ABSTRACT

Title of dissertation: ALGORITHMS AND EVALUATION FOR OBJECT DETECTION AND TRACKING IN COMPUTER VISION

Kyungnam Kim, Doctor of Philosophy, 2005

Dissertation directed by: Professor Larry Davis
Department of Computer Science

Vision-based object detection and tracking, especially for video surveillance applications, is studied from algorithms to performance evaluation. This dissertation is composed of four topics: (1) Background Modeling and Detection, (2) Performance Evaluation of Sensitive Target Detection, (3) Multi-view Multi-target Multi-Hypothesis Segmentation and Tracking of People, and (4) A Fine-Structure Image/Video Quality Measure.

First, we present a real-time algorithm for foreground-background segmentation. It allows us to capture structural background variation due to periodic-like motion over a long period of time under limited memory. Our codebook-based representation is efficient in memory and speed compared with other background modeling techniques. Our method can handle scenes containing moving backgrounds or illumination variations, and it achieves robust detection for different types of videos. In addition to the basic algorithm, three features improving the algorithm are presented - Automatic Parameter Estimation, Layered Modeling/Detection and Adaptive Codebook Updating.

Second, we introduce a performance evaluation methodology called Perturbation Detection Rate (PDR) analysis for measuring performance of foreground-background segmentation. It does not require foreground targets or knowledge of foreground distributions. It measures the sensitivity of a background subtraction algorithm in detecting possible low contrast targets against the background as a function of contrast. We compare four background subtraction algorithms using the methodology.

Third, a multi-view multi-hypothesis approach to segmenting and tracking multiple persons on a ground plane is proposed. The tracking state space is the set of ground points of the people being tracked. During tracking, several iterations of segmentation are performed using information from human appearance models and ground plane homography. Two innovations are made in this chapter - (1) To more precisely locate the ground location of a person, all center vertical axes of the person across views are mapped to the top-view plane to find the intersection point. (2) To tackle the explosive state space due to multiple targets and views, iterative segmentation-searching is incorporated into a particle filtering framework. By searching for people's ground point locations from segmentations, a set of a few good particles can be identified, resulting in low computational cost. In addition, even if all the particles are away from the true ground point, some of them move towards the true one through the iterated process as long as they are located nearby.

Finally, an objective no-reference measure is presented to assess fine-structure image/video quality. The proposed measure using local statistics reflects image degradation well in terms of noise and blur.

ALGORITHMS AND EVALUATION FOR OBJECT DETECTION AND TRACKING IN COMPUTER VISION

by

Kyungnam Kim

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

Advisory Committee:

Professor Larry Davis, Chair/Advisor
Professor David Jacobs
Professor Amitabh Varshney
Dr. David Doermann
Professor Sung W. Lee

Dedication

To My Family and Teachers

ACKNOWLEDGMENTS

The research reported herein could not have been completed without the guidance and help of many individuals. I would like to single out the following people for their influence over the last two formative years.

First and foremost I wish to convey my deep sense of gratitude to my advisor, Professor Larry S. Davis, for his constant help and encouragement all through the period of this work. It was because of him that my graduate studies were so enjoyable and so intellectually rewarding.

I am also grateful to my mentor, David Harwood. His extraordinary expertise and guide in the course of this research has been significant.

Thanks are due to Professor David Jacobs, Professor Amitabh Varshney, Dr. David Doermann and Professor Sung W. Lee for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript. I am also indebted to Professor Rama Chellappa who was a committee member of my thesis proposal and Professor Ramani Duraswami who friendly helped me in many ways.

My years at the University or Maryland have been highly rewarding. I consider myself lucky to be part of this institute. I acknowledge the encouragement of my colleagues through their remarks and observations on this work. Thanks are especially due to the Korean Graduate Vision and AI Research Group members, Bo-

hyung Han, Kyongil Yoon, Seong-wook Joo, Hyoungjune Yi, and Minkyoung Cho. Our informal group seminars and discussion were so valuable to me. In addition, my colleagues at the Computer Vision Laboratory have enriched my graduate life in many ways and deserve a special mention - Thanarat H Chalidabhongse, Harsh Nanda, Anurag Mittal, Changjiang Yang, Sernam Lim, Motilal Agrawal, Ahmed Elgammal, Ali Zandifar, Chiraz BenAbdelkader, Liang Zhao and Vinay Shet. My co-worker Thanarat helped me to start off this research work. I record my appreciation of the direct and indirect assistance rendered to me by the people in the Department of Computer Science.

I owe my deepest thanks to my parents and parents-in-law who have always stood by me and guided me through my career, and have pulled me through against impossible odds at times. Words cannot express the gratitude I owe them. Last but not least, I would like to thank my wife Jong-Su Paek and my son Asher for their patience, love, and unconditional support during my graduate work.

TABLE OF CONTENTS

List of Figures vii

1 Introduction 1
1.1 Motivations and Related Work . . . 1
1.1.1 Background Modeling and Detection . . . 1
1.1.2 Performance Evaluation of Sensitive Target Detection . . . 3
1.1.3 Multi-view Multi-target Multi-Hypothesis Segmentation and Tracking of People . . . 4
1.1.4 A Fine-Structure Image/Video Quality Measure . . . 7
1.2 Contributions . . . 9
1.2.1 Background Modeling and Detection . . . 9
1.2.2 Performance Evaluation of Sensitive Target Detection . . . 9
1.2.3 Multi-view Multi-target Multi-Hypothesis Segmentation and Tracking of People . . . 10
1.2.4 A Fine-Structure Image/Video Quality Measure . . . 10
1.3 Thesis Organization . . . 11

2 Background Modeling and Detection 13
2.1 Overview . . . 13
2.2 Background modeling and detection . . . 14
2.2.1 Construction of the initial codebook . . . 14
2.2.2 Maximum Negative Run-Length . . . 17
2.2.3 Color and Brightness . . . 19
2.2.4 Foreground Detection . . . 22
2.2.5 Review of multimode modeling techniques . . . 23
2.3 Detection Results and Comparison . . . 25
2.4 Automatic Parameter Estimation - ϵ_1 and ϵ_2 . . . 30
2.5 Layered modeling and detection - Model maintenance . . . 31
2.5.1 Overview . . . 31
2.5.2 Model updating for background changes . . . 33
2.5.3 Issues for Background Updating . . . 35
2.5.4 Experimental results - examples . . . 38
2.6 Adaptive codebook updating - detection under global illumination changes . . . 41
2.7 Conclusion and Discussion . . . 42

3 Performance Evaluation of Sensitive Target Detection 46
3.1 PDR - A performance evaluation method . . . 46
3.1.1 Concept . . . 46
3.1.2 PDR algorithm . . . 47
3.2 Results . . . 48
3.2.1 Tested algorithms and experiment setups . . . 48
3.2.2 Indoor and outdoor videos . . . 50

3.2.3 Detection sensitivity - a real example . . . 52
3.2.4 Multiple moving backgrounds . . . 53
3.2.5 Different color models . . . 55
3.3 Conclusions and future work . . . 56
3.3.1 Conclusions . . . 56
3.3.2 Future Work . . . 58

4 Multi-view Multi-target Multi-Hypothesis Segmentation and Tracking of People 60
4.1 Human appearance model . . . 60
4.2 Multi-camera Multi-person Segmentation and Tracking . . . 62
4.2.1 Foreground segmentation . . . 62
4.2.2 Model initialization and update . . . 65
4.2.3 Multi-view integration . . . 66
4.3 Extension to Multi-hypothesis Tracker . . . 70
4.3.1 Overview of particle filter . . . 70
4.3.2 State space and dynamics . . . 72
4.3.3 Observation . . . 72
4.3.4 The final algorithm . . . 74
4.4 Experiments . . . 76
4.5 Conclusions . . . 80

5 A Fine-Structure Image/Video Quality Measure using Local Statistics 81
5.1 Video properties: Q1 - Q4 . . . 81
5.2 Fine-structure image/video quality measure . . . 82
5.3 Experimental Results . . . 87
5.4 Conclusions and Future Work . . . 89

6 Conclusions 91

Bibliography 94

LIST OF FIGURES

2.1 Example showing how MNRL is used. 17

2.2 The distributions of 4 pixel values of the color-chart image sequence having illumination changes over time 20

2.3 The proposed color model - a separate evaluation of color distortion and brightness distortion. 21

2.4 Detection results on a compressed video 25

2.5 Detection results on multiple moving backgrounds 26

2.6 Detections results on training of non-clean backgrounds 27

2.7 Detections results on very long-time backgrounds 28

2.8 The overview of our approach with short-term background layers: the foreground and the short-term backgrounds can be interpreted in a different temporal order. The diagram items in dotted line, such as Tracking, are added to complete a video surveillance system. 34

2.9 Two cases of changed backgrounds. The case (2) shows almost homogeneous neighborhoods over the boundary while different colors are observed on the opposite sides along the boundary direction in the case (1). 37

2.10 Layered modeling and detection - A woman placed a box on the desk and then the box has been absorbed into the background model as short-term. Notice that the paper on the desk was displaced by the box and then also became a short-term background layer. Then a purse is put in front of the box. The purse is detected against both the box and the desk. 38

2.11 The leftmost column: original images, the middle column: color-labelled short-term backgrounds, the rightmost column: detected foreground. The video shows that a man parks his car on the lot and takes out two boxes. He walks away to deliver them. 39

2.12 The sample frames are shown in the order of time along with short-term background layers in the second column. (a): A bag is placed by somebody and left unattended. A short-term background layer is formed. It is still memorized as a layer after the door was closed and then opened. (b): While several people walk in and out the office, a bag has been left without any attention. Even with severe occlusion by walking people, the bag stands out as a layer. 40

2.13 Results of adaptive codebook updating for detection under global illumination changes. Detected foregrounds on the frame 1105 are labelled with green color. 43

3.1 The sample empty-frames of four videos used in the experiments . . . 49

3.2 PDR for 'indoor office' video in Figure 3.1(a) 51

3.3 PDR for 'outdoor woods' video in Figure 3.1(b) 51

3.4 Sensitive detection at small delta 52

3.5 PDR for 'red-brick wall' video in Figure 3.1(c) 53

3.6 PDR for window on moving background (Figure 3.1(d)) 54

3.7 PDR of different color models for the video in Figure 3.1(a) 55

4.1 Illustration of appearance model for each body part. h_k and w_k are measured relative to the full height of the person. 64

4.2 Detection of persons for initialization of the appearance model. The bounding boxes in the figures were created when the blobs are isolated before. 65

4.3 Wrong ground points were detected due to the broken segmentation and the shadow under the feet. 67

4.4 All vertical axes of a person across views intersect at (or are very close to) a single point when mapped to the top-view. 69

4.5 The proposed system tracks the ground positions of people over nearly 1000 frames. A small ball marker are overlaid on the resultant images of the frame 292 for easy finding of the camera orientations. Four representative frames are selected, which show the difficulty of tracking due to severe occlusion. Additionally, persons 2 and 3 are similar in their appearance colors. Note that, in the figures of 'vertical axes', the axis of a severely occluded person does not account for localization of the ground point. 77

4.6 Comparison on three methods: While the deterministic search with a single hypothesis (persons 2 and 4 are good) and the general particle filter (only person 3 is good) fail in tracking all the persons correctly, our proposed method succeeds with a minor error. The view 2 was only shown here. 78

4.7 Segmentation results: Our method provides generally accurate blobs. The segmentation results which reflect occlusion can be obtained through the Bayesian pixel classification (Eq.4.2) based on the geometric constraints (Eq.4.4) determined by the locations of the ground points. 79

5.1 (left) 3×3 neighborhood with 4 center-symmetric pairs of pixels, (right) original space station image 83

5.2 CSAC histograms of original, noise, and blur images 84

5.3 two 3×3 boxes and their average. 84

5.4 Cumulative FIQ histograms - GaussianNoise 86

5.5 Cumulative FIQ histograms - GaussianBlur 87

Chapter 1

Introduction

1.1 Motivations and Related Work

1.1.1 Background Modeling and Detection

The capability of extracting moving objects from a video sequence captured using a static camera is a typical first step in visual surveillance. A common approach for discriminating moving objects from the background is detection by background subtraction (BGS). The idea of background subtraction is to subtract or difference the current image from a reference background model. The subtraction leaves only non-stationary or new objects.

The simplest background model assumes that the intensity values of a pixel can be modeled by a Gaussian distribution $N(\mu, \sigma^2)$. This basic model is used in [1, 2]. However, a single Gaussian model cannot handle multiple backgrounds, like waving trees. The generalized mixture of Gaussians (MOG) has been used to model complex, non-static backgrounds [3, 5].

The MOG has some disadvantages. Backgrounds having fast variations cannot be modeled with just a few Gaussians accurately, so fail to provide sensitive detection [9]. In addition, depending on the learning rate to adapt to background changes, MOG faces trade-off problems. For a low learning rate, it produces a wide model not

able to detect a sudden change to the background. If the model adapts too quickly, slowly moving foregrounds will be absorbed into the background model, resulting in a high false negative rate. This is the foreground aperture problem described in [10]. To overcome these problems, a non-parametric technique estimating the probability density function at each pixel from many samples using Kernel density estimation technique was developed. It is able to adapt very quickly to changes in the background process and to detect targets with high sensitivity [9].

These pixel-based techniques assume that the time series of observation is independent at each pixel. In contrast, some researchers employ a region- or frame-based approach by segmenting an image into regions or by refining low-level classification obtained at the pixel level [10, 5, 8].

To deal with global and local illumination changes such as shadows and highlights, algorithms generally employ normalized colors. These techniques typically work poorly in dark areas of the image. This problem is addressed in [17]. This uncertainty makes the detection in dark regions unstable. We present color metrics to reduce the uncertainty and handle shadows and highlights effectively.

MOG and the non-parametric technique in [9] cannot be used when long-time periods are needed to sufficiently sample the background - for example when there is significant wind load on vegetation - due mostly to memory constraints. We construct a highly compressed background model that addresses that problem.

Existing algorithms usually require uncompressed, high-resolution video. With the rapid growth of Internet and multimedia communications, the demand for robust algorithms that can analyze compressed data transmitted over lossy channels has

been increasing. We desire an algorithm that is independent of the image sources or encoders. In other words, the algorithm should work well on both uncompressed and compressed videos regardless of the compression standards. Most existing background subtraction algorithms fail to work with low-bandwidth compressed videos mainly due to spatial block compression that causes block artifacts (see Fig. 2.4(b)), and temporal block compression that causes abnormal distribution of encoding (random spikes). Our new method is robust with respect to image quality.

Our codebook (CB) background subtraction algorithm was intended to sample values over long times, without making parametric assumptions. It might be applicable to compressed video, which often has unusual, discontinuous distributions, as well as to uncompressed video. Mixed backgrounds can be modeled by multiple codewords, while brightness and color are separated.

1.1.2 Performance Evaluation of Sensitive Target Detection

When comparing BGS algorithms [41] or evaluating computer vision systems [43, 44], ROC analysis is often employed when there are known background and foreground (target) distributions. ROC curves display the detection sensitivity for detecting a particular foreground against a particular background, but the methodology has some disadvantages for evaluating BGS algorithms. There are as many ROC curves as there are possible different foreground targets. In addition, it requires considerable experimentation and ground-truth evaluation to obtain accurate false alarm rates (FA) and the miss detection rates (MD). Most important, in typical video surveillance applications, we usually are given a background scene for a fixed

camera, but we do not or can not know what might possibly move in the scene as foreground objects.

The perturbation method presented here, called perturbation detection rate (PDR) analysis, measures the sensitivity of a BGS algorithm without assuming knowledge of the actual foreground distribution. Rather, it measures the detection of a variable, small ("just-noticeable") difference from the background, obtaining a foreground distribution by assuming that the foreground might have a distribution locally similar in form to the background, but shifted or perturbed. The detection is measured as a function of contrast, the magnitude of the shift or perturbation in uniform random directions in RGB.

1.1.3 Multi-view Multi-target Multi-Hypothesis Segmentation and Tracking of People

Tracking and segmenting people in cluttered or complex situations is a challenging visual surveillance problem since the high density of objects results in occlusion and lack of visibility. Elgammal and Davis [64] presented a general framework which uses maximum likelihood estimation and occlusion reasoning to obtain the best arrangement for people that yields an observed segmentation for the foreground region. However, when the number of people are increased, the complexity of possible arrangement hypotheses increases dramatically and only a heuristic method to limit the hypothesis space is suggested. To handle more people in a crowded scene, Zhao and Nevatia [53] described a model-based segmentation approach to segment individual humans in a high-density scene using a Markov chain Monte Carlo method.

When a single camera is not sufficient to detect and track objects due to limited visibility or occlusion, multiple cameras can be employed. There are a number of papers which address detection and tracking using overlapping or non-overlapping multiple views. In [49], approaches for continuous detection and tracking by multiple, stationary or moving cameras are described based on Tensor Voting for continuous trajectories and a joint probability model for integrating all the information across cameras. Tracking across multiple cameras with non-overlapping views is also addressed in [50]. Without the requirement of inter-camera calibration, object correspondence across cameras is established by a Bayesian framework incorporating multiple cues such as location of exit/entrances.

On the other hand, M_2Tracker [63], which is similar to our work, used a region-based stereo algorithm to find 3D points inside an object, and Bayesian pixel classification with occlusion analysis to segment people occluded in different levels of crowd density. Unlike M_2Tracker's requirement of having calibrated stereo pairs of cameras, we do not require strong calibration, but only a ground plane homography. For outdoor cameras, it is practically very difficult to accurately calibrate them, so that 3D points at a large distance from the camera cannot be measured accurately.

Our goal is to 'segment' and 'track' people on a ground plane viewed from multiple overlapping cameras. Although the condition of overlap is not necessary, when the density of targets is high, i.e., crowded, it is much better to have different views of the targets to resolve the occlusion problem. The application of our work would be monitoring crowed spaces like building entrances, stores, casinos, subway stations etc. Human appearance models are used to segment foreground pixels obtained

from background subtraction. Given the estimated ground points of the people, the occlusion order in each view is determined and used for segmentation. Then each segmented blob of a person across views is related by a ground plane homography to locate a final ground point of the person. The ground point is estimated on the top-view reconstruction so that our tracking framework does not need to compare every pair of views. We designed a method of view integration which robustly works on imperfectly segmented blobs. This is quite useful because background subtraction and segmentation are not always reliable due to noise, illumination changes, etc.

To make the tracker robust, multiple hypothesis trackers, such as *particle filter* [56], are widely used since they provide a robust framework with simplicity, generality and success in a wide range of challenging applications [61, 60]. However, our work is about segmenting and tracking multiple targets from multiple views. As the number of targets and views increase, the state space of combination of targets' states also increase exponentially. Additionally the observation processes for visual tracking are typically computationally expensive. Previous research has tried to solve this state space explosion issue. For example, annealed particle filtering [57] has been used to perform articulated body motion tracking, which involves a large number of degrees of freedom, by searching high dimensional configuration spaces effectively using a continuation principle. [45] presents a computational method called data-driven Markov chain Monte Carlo for image segmentation in the Bayesian statistical framework. It utilizes data-driven (bottom-up) techniques, such as clustering and edge detection, to compute importance proposal probabilities, which drive the

Markov chain dynamics and achieve large speedup in comparison to the traditional diffusion methods. Sullivan and Rittscher [58] proposed an algorithm to incorporate the strength of both particle filtering (a random search guided by a stochastic model) and variational approaches (a deterministic and data-driven search minimizing a cost function [51]). Similarly, a hand tracking algorithm that combines particle filtering and mean-shift was presented in [59] to realize more efficient sampling by shifting samples to their neighboring modes to overcome the degeneracy problem and require fewer particles to maintain multiple hypotheses.

We designed our tracker using the same philosophy. Nevertheless, in our multi-camera multi-target tracking framework, it is hard to obtain analytic or direct solutions for searching the state space. We extend the framework to a multi-hypothesis version using particle filtering. Each hypothesis is efficiently refined by the multi-view segmentation results to maintain only optimal samples, resulting in low computational costs.

1.1.4 A Fine-Structure Image/Video Quality Measure

Distortion is introduced in images and video through various processes such as acquisition, transmission and compression. There are many ways to measure image/video quality by objective or subjective assessment. Subjective evaluations [65] are expensive and time-consuming. It is impossible to implement them into automatic real-time systems. Subjective measurements can be used to validate the usefulness of objective measurements by showing strong correlations.

Many objective image quality measures have been proposed from simple mean

squared error (MSE) metrics to some measures incorporating elements of human visual perception. It is well-known that MSE is suitable to describe the subjective degradation perceived by a viewer. To overcome the limitations of MSE, other measures mimick the human visual system. For example, video quality metrics based on the Standard Spatial Observer were presented in [66]. A power spectrum approach which does not require imaging a specific pattern or a constant scene was reported in [67]. Recently, assuming that human visual perception is highly adapted to extract structural information from scenes, a framework for quality assessment based on the degradation of structural information was proposed in [68]. The video quality expert group (VQEG) [69] is working on validating and standardizing video quality metrics for television and multimedia applications. A number of objective quality measures are evaluated and categorized in [70, 71].

Some measures [67, 72, 73] estimate the quality of coded video or images without a reference quality standard such as an original perfect image. In many situations, we cannot guarantee that original sources of imaging or degradation processes are available. No-reference metrics are not relative to the original but are absolute values for a test image or video.

Our quality measure is no-reference objective metric employing local statistics to assess image/video quality. While most techniques measure degradation of compressed images or video, ours takes any input image without reference and assesses a quality measure related to performance of video surveillance tasks. Our measurement may not have a strong correlation with subjective evaluation provided by a human observer.

1.2 Contributions

1.2.1 Background Modeling and Detection

The key contributions of our background subtraction algorithm described in this dissertation are in

- resistance to artifacts of acquisition, digitization and compression.
- capability of coping with local and global illumination changes.
- adaptive and compressed background model that can capture structural background motion over a long period of time under limited memory. This allows us to encode moving backgrounds or multiple changing backgrounds.
- unconstrained training that allows moving foreground objects in the scene during the initial training period.
- automatic parameter estimation
- layered modeling and detection allowing us to have multiple layers of background representing different depths

1.2.2 Performance Evaluation of Sensitive Target Detection

PDR analysis has two advantages over the commonly used ROC analysis: (1) It does not depend on knowing foreground distributions, (2) It does not need the presence of foreground targets in the video in order to perform the analysis, while this is required in the ROC analysis. Because of these considerations, PDR analysis provides practical general information about the sensitivity of algorithms applied to a given video scene over a range of parameters and FA-rates. In ROC curves, we obtain one detection rate for a particular FA-rate for a particular foreground and contrast.

1.2.3 Multi-view Multi-target Multi-Hypothesis Segmentation and Tracking of People

Our framework for segmenting and tracking people on a ground plane makes important contributions:

1. To more precisely locate the ground location of a person, all center vertical axes of the person across views are mapped to the top-view plane (rather than compared within a pair of views) to find the intersection point.

2. To tackle the explosive state space due to multiple targets and views, an iterative segmentation-searching is incorporated into a particle filtering framework. By searching for a person's ground point from segmentation, a set of a few good particles can be identified, resulting in low computational costs. In addition, even if all the particles are away from the true ground point, some of them move towards the true one as long as they are located nearby.

1.2.4 A Fine-Structure Image/Video Quality Measure

Our quality measure is no-reference objective metric employing local statistics to assess image/video quality. While most techniques measure degradation of compressed images or video, ours takes any input image without reference and assesses a quality measure. The proposed measure reflects image degradation well in terms of noise and blur.

1.3 Thesis Organization

This thesis is organized in three major topics as briefly described in the previous section.

In Chapter 2, we describe the codebook construction algorithm and the color and brightness metric, used for detection. We show that the method is suitable for both stationary and moving backgrounds in different types of scenes, and applicable to compressed videos such as MPEG. Important improvements to the above algorithm are presented - Automatic Parameter Estimation, Layered Modeling/Detection and Adaptive Codebook Updating. Finally, conclusion and discussion are presented in the last section.

In Chapter 3, we describe the performance evaluation technique, PDR analysis. It measures the sensitivity of a BGS algorithm without assuming knowledge of the actual foreground distribution. Then the experimental results for the four background subtraction algorithms are presented along with some discussions. Conclusions and future work are given in the last section.

In Chapter 4, a multi-view multi-target multi-hypothesis tracker is proposed. It segments and tracks people on a ground plane. Human appearance models are used to segment foreground pixels obtained from background subtraction. We developed a method to effectively integrate segmented blobs across views on a top-view reconstruction, with a help of ground plane homography. The multi-view tracker is extended efficiently to a multi-hypothesis framework (M^3Tracker) using particle filtering.

In Chapter 5, we describe four video properties to be considered for video surveillance applications. Then, our fine-structure image/video quality measure is detailed. Experimental results are presented along with the performance of a background subtraction algorithm.

In the last chapter, we summarize the work that has been done in the dissertation and discuss possible future directions of research.

Chapter 2

Background Modeling and Detection

2.1 Overview

Our codebook (CB) background subtraction algorithm was intended to sample values over long times, without making parametric assumptions. Mixed backgrounds can be modeled by multiple codewords. The key features of the algorithm are

- an adaptive and compact background model that can capture structural background motion over a long period of time under limited memory. This allows us to encode moving backgrounds or multiple changing backgrounds.
- the capability of coping with local and global illumination changes.
- unconstrained training that allows moving foreground objects in the scene during the initial training period.
- layered modeling and detection allowing us to have multiple layers of background representing different background layers

In Section 2.2, we describe the codebook construction algorithm and the color and brightness metric, used for detection. We show, in Section 2.3, that the method is suitable for both stationary and moving backgrounds in different types of scenes, and applicable to compressed videos such as MPEG. Important improvements to the above algorithm are presented in Section 2.42.5, and 2.6 - Automatic Parameter Estimation, Layered Modeling/Detection and Adaptive Codebook Updating.

Finally, conclusion and discussion are presented in the last section.

2.2 Background modeling and detection

The CB algorithm adopts a quantization/clustering technique, inspired by Kohonen [18, 19], to construct a background model from long observation sequences. For each pixel, it builds a codebook consisting of one or more codewords. Samples at each pixel are clustered into the set of codewords based on a color distortion metric together with brightness bounds. Not all pixels have the same number of codewords. The clusters represented by codewords do not necessarily correspond to single Gaussian or other parametric distributions. Even if the distribution at a pixel were a single normal, there could be several codewords for that pixel. The background is encoded on a pixel by pixel basis.

Detection involves testing the difference of the current image from the background model with respect to color and brightness differences. If an incoming pixel meets two conditions, it is classified as background - (1) The color distortion to some codeword is less than the detection threshold, and (2) its brightness lies within the brightness range of that codeword. Otherwise, it is classified as foreground.

2.2.1 Construction of the initial codebook

The algorithm is described for color imagery, but it can also be used for gray-scale imagery with minor modifications. Let $\mathcal{X}$ be a training sequence for a single pixel consisting of N RGB-vectors: $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$. Let $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_L\}$ represent the codebook for the pixel consisting of L codewords. Each pixel has a

different codebook size based on its sample variation.

Each codeword $\mathbf{c}_i$, $i = 1 \ldots L$, consists of an RGB vector $\mathbf{v}_i = (\bar{R}_i, \bar{G}_i, \bar{B}_i)$ and a 6-tuple $\mathbf{aux}_i = \langle \check{I}_i, \hat{I}_i, f_i, \lambda_i, p_i, q_i \rangle$. The tuple $\mathbf{aux}_i$ contains intensity (brightness) values and temporal variables described below.

$\check{I}, \hat{I}$: the *min* and *max* brightness, respectively, of all pixels assigned to this codeword;

f : the *frequency* with which the codeword has occurred;

λ : the *maximum negative run-length* (MNRL) defined as the longest interval during the training period that the codeword has NOT recurred;

p, q : the *first* and *last* access times, respectively, that the codeword has occurred.

In the training period, each value, $\mathbf{x}_t$, sampled at time t is compared to the current codebook to determine which codeword $\mathbf{c}_m$ (if any) it matches (m is the matching codeword's index). We use the matched codeword as the sample's encoding approximation. To determine which codeword will be the best match, we employ a color distortion measure and brightness bounds. The detailed algorithm is given below.

Algorithm for Codebook Construction

I. $L \leftarrow 0$[1], $\mathcal{C} \leftarrow \emptyset$ (empty set)

II. **for** t=1 to N **do**

i. $\mathbf{x}_t = (R, G, B)$, $I \leftarrow \sqrt{R^2 + G^2 + B^2}$

ii. Find the codeword $\mathbf{c}_m$ in $\mathcal{C} = \{\mathbf{c}_i | 1 \leq i \leq L\}$ matching to $\mathbf{x}_t$ based on two conditions (a) and (b).

(a) $colordist(\mathbf{x}_t, \mathbf{v}_m) \leq \epsilon_1$

(b) $brightness(I, \langle \check{I}_m, \hat{I}_m \rangle) =$ **true**

iii. If $\mathcal{C} = \emptyset$ or there is no match, then $L \leftarrow L + 1$. Create a new codeword $\mathbf{c}_L$ by setting

- $\mathbf{v}_L \leftarrow (R, G, B)$
- $\mathbf{aux}_L \leftarrow \langle I, I, 1, t-1, t, t \rangle$.

iv. Otherwise, update the matched codeword $\mathbf{c}_m$, consisting of $\mathbf{v}_m = (\bar{R}_m, \bar{G}_m, \bar{B}_m)$ and $\mathbf{aux}_m = \langle \check{I}_m, \hat{I}_m, f_m, \lambda_m, p_m, q_m \rangle$, by setting

- $\mathbf{v}_m \leftarrow (\frac{f_m \bar{R}_m + R}{f_m + 1}, \frac{f_m \bar{G}_m + G}{f_m + 1}, \frac{f_m \bar{B}_m + B}{f_m + 1})$
- $\mathbf{aux}_m \leftarrow \langle\ min\{I, \check{I}_m\}, max\{I, \hat{I}_m\}, f_m + 1, max\{\lambda_m, t - q_m\}, p_m, t\ \rangle$.

end for

III. For each codeword $\mathbf{c}_i$, $i = 1 \ldots L$, wrap around λ_i by setting $\lambda_i \leftarrow max\{\lambda_i, (N - q_i + p_i - 1)\}$.

The two conditions (a) and (b) in the Step II-ii, detailed in Eq.2.2,2.3 later, are satisfied when the pure colors of $\mathbf{x}_t$ and $\mathbf{c}_m$ are close enough and the brightness of $\mathbf{x}_t$ lies between the acceptable brightness bounds of $\mathbf{c}_m$. Instead of finding the nearest neighbor, we just find the first codeword to satisfy these two conditions. ϵ_1 is the sampling threshold (bandwidth). One way to improve the speed of the algorithm is to relocate the most recently updated codeword to the front of the

[1] $\leftarrow$ means assignment

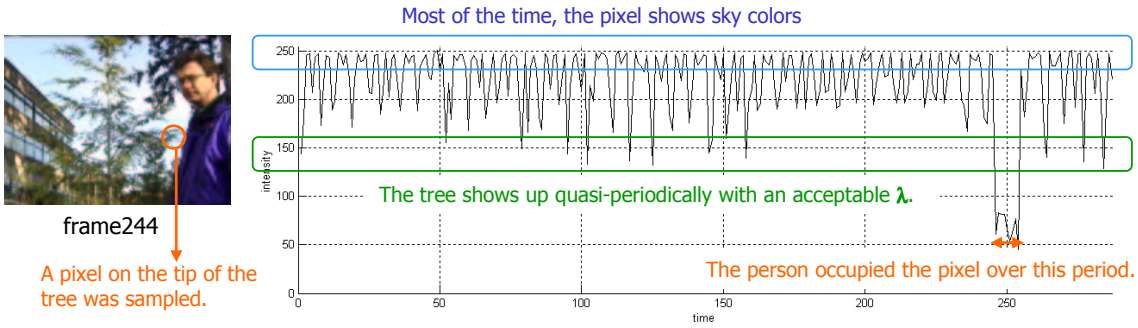


Figure 2.1: Example showing how MNRL is used.

codebook list. Most of the time, the matched codeword was the first codeword thus relocated, making the matching step efficient.

Note that reordering the training set almost always results in codebooks with the same detection capacity. Reordering the training set would require maintaining all or a large part of it in memory. Experiments show that one-pass training is sufficient. Retraining or other simple "batch" processing methods do not affect detection significantly.

2.2.2 Maximum Negative Run-Length

We refer to the codebook obtained from the previous step as the fat codebook. It contains all the codewords that represent the training image sequence, and may include some moving foreground objects and noise.

In the temporal filtering step, we refine the fat codebook by separating the codewords that might contain moving foreground objects from the true background codewords, thus allowing moving foreground objects during the initial training period. The true background, which includes both static pixels and moving background pixels, usually is quasi-periodic (values recur in a bounded period). This motivates

the temporal criterion of MNRL (λ), which is defined as the maximum interval of time that the codeword has not recurred during the training period. For example, as shown in Fig.2.1, a pixel on the tip of the tree was sampled to plot its intensity variation over time. The codeword of sky-color has a very small λ, around 15, and that of tree-color has 100. However, the codeword of the person's body has a very large λ, 280.

Let $\mathcal{M}$ and $T_{\mathcal{M}}$ denote the background model (which is a refined codebook after temporal filtering) and the threshold value respectively. Usually, $T_{\mathcal{M}}$ is set equal to half the number of training frames, $\frac{N}{2}$.

$$\mathcal{M} = \{\mathbf{c}_m | \quad \mathbf{c}_m \in \mathcal{C} \quad \wedge \quad \lambda_m \leq T_{\mathcal{M}}\} \tag{2.1}$$

Codewords having a large λ will be eliminated from the codebook by Eq.2.1. Even though one has a large frequency 'f', its large λ means that it is mostly a foreground event which was stationary only for that period f. On the other hand, one having a small f and a small λ could be a rare background event occurring quasi-periodically. We can use λ as a feature to discriminate the actual background codewords from the moving foreground codewords. If $T_{\mathcal{M}} = \frac{N}{2}$, all the codewords should recur at least every $\frac{N}{2}$ frames. We note that we also experimented with the combination of the frequency f and λ, but that λ alone performs almost the same as that combination.

Experiments on many videos reveal that only 6.5 codewords per pixel (on average) are required for the background acquisition in order to model 5 minutes of outdoor video captured at 30 frames per second. By contrast, indoor videos are

simpler, having one or two background values nearly everywhere. This reasonable number of codewords means that our method achieves a high compression of the background model. This allows us to capture variable moving backgrounds over a very long period of training time with limited memory.

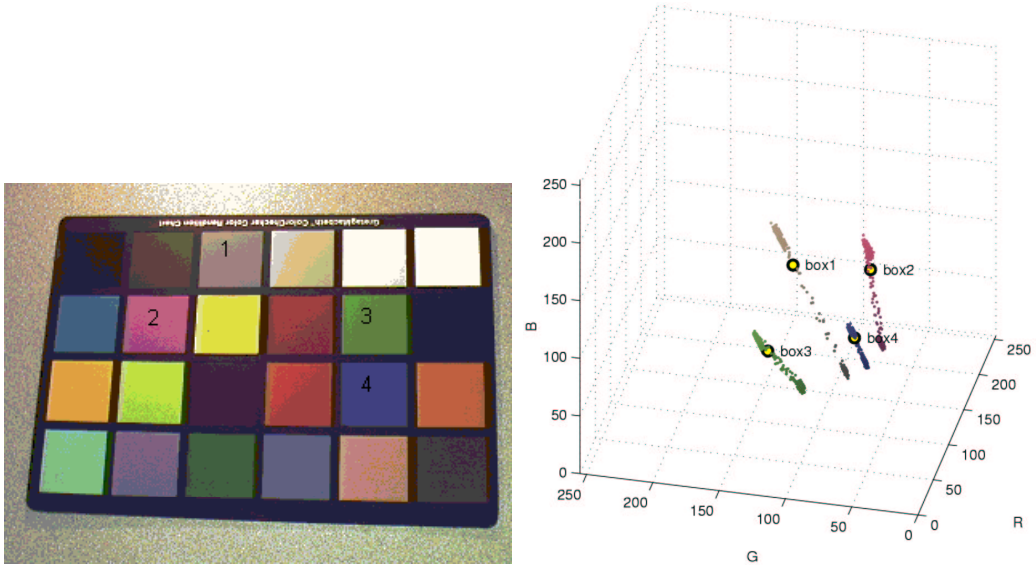
2.2.3 Color and Brightness

To deal with global and local illumination changes such as shadows and highlights, algorithms generally employ normalized colors (color ratios). These techniques typically work poorly in dark areas of the image. The dark pixels have higher uncertainty[2] than the bright pixels, since the color ratio uncertainty is related to brightness. Brightness should be used as a factor in comparing color ratios. This uncertainty makes the detection in dark regions unstable. The false detections tend to be clustered around the dark regions. This problem is discussed in [17].

Hence, we observed how pixel values change over time under lighting variation. Fig.2.2(b) shows the pixel value distributions in the RGB space where 4 representative pixels are sampled from the image sequence of the color-chart in Fig.2.2(a). In the sequence captured in a lab environment, the illumination changes over time by decreasing or increasing the light strength to make the pixel values darker or brighter. The pixel values are mostly distributed in elongated shape along the axis going toward the origin point $(0, 0, 0)$.

Based on this observation, we developed a color model depicted in Fig.2.3 to

[2]Consider two pairs of two color values at the same Euclidean distance in RGB space - $\langle 10, 10, 10\rangle$ and $\langle 9, 10, 11\rangle$ for dark pixels, $\langle 200, 200, 200\rangle$ and $\langle 199, 200, 201\rangle$ for bright pixels. Their distortions in normalized colors are $\frac{2}{30} = \frac{|10-9|+|10-10|+|10-11|}{30}$ and $\frac{2}{200} = \frac{|200-199|+|200-200|+|200-201|}{200}$ respectively.



(a) original color-chart image (b) 3D plot of pixel distributions

Figure 2.2: The distributions of 4 pixel values of the color-chart image sequence having illumination changes over time

perform a separate evaluation of color distortion and brightness distortion. The motivation of this model is that background pixel values lie along the principal axis of the codeword along with the low and high bound of brightness, since the variation is mainly due to brightness. When we have an input pixel $\mathbf{x}_t = (R, G, B)$ and a codeword $\mathbf{c}_i$ where $\mathbf{v}_i = (\bar{R}_i, \bar{G}_i, \bar{B}_i)$,

$$\|\mathbf{x}_t\|^2 = R^2 + G^2 + B^2,$$

$$\|\mathbf{v}_i\|^2 = \bar{R}_i^2 + \bar{G}_i^2 + \bar{B}_i^2,$$

$$\langle \mathbf{x}_t, \mathbf{v}_i \rangle^2 = (\bar{R}_i R + \bar{G}_i G + \bar{B}_i B)^2.$$

The color distortion δ can be calculated by

$$\begin{aligned} p^2 &= \|\mathbf{x}_t\|^2 \cos^2\theta = \frac{\langle \mathbf{x}_t, \mathbf{v}_i \rangle^2}{\|\mathbf{v}_i\|^2} \\ colordist(\mathbf{x}_t, \mathbf{v}_i) &= \delta = \sqrt{\|\mathbf{x}_t\|^2 - p^2}. \end{aligned} \tag{2.2}$$

Our color distortion measure can be interpreted as a brightness-weighted version in the normalized color space. This is equivalent to geometrically rescaling

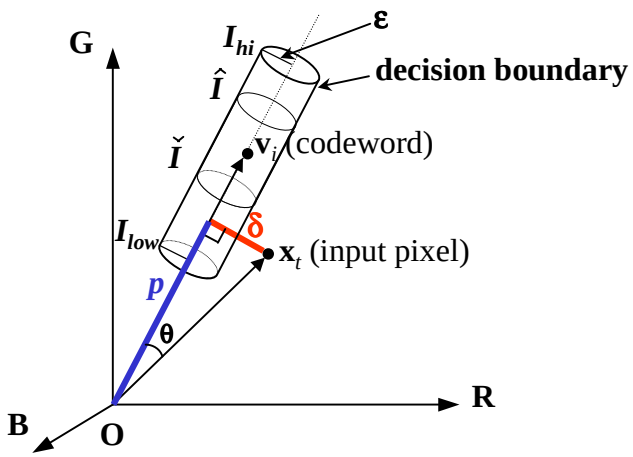


Figure 2.3: The proposed color model - a separate evaluation of color distortion and brightness distortion.

(normalizing) a codeword vector to the brightness of an input pixel. This way, the brightness is taken into consideration for measuring the color distortion, and we avoid the instability of normalized colors.

To allow for brightness changes in detection, we store $\check{I}$ and $\hat{I}$ statistics, which are the min and max brightness of all pixels assigned to a codeword, in the 6-tuple defined in Section 2.2.1. We allow the brightness change to vary in a certain range that limits the shadow level and highlight level. The range is $[I_{low}, I_{hi}]$, for each codeword, defined as

$$I_{low} = \alpha\hat{I}, \quad I_{hi} = min\{\beta\hat{I}, \frac{\check{I}}{\alpha}\}.$$

where $\alpha < 1$ and $\beta > 1$. Typically, α is between 0.4 – 0.7[3], and β is between 1.1 – 1.5[4]. This range $[I_{low}, I_{hi}]$ becomes a stable range during codebook updating. The

[3]These typical values are obtained from experiments. 0.4 allows large brightness bounds, but 0.7 gives tight bounds

[4]β is additionally used for limiting I_{hi} since shadows (rather than highlights) are observed in most cases

logical brightness function in Section 2.2.1 is defined as

$$brightness(I, \langle \check{I}, \hat{I} \rangle) = \begin{cases} \textbf{true} & \text{if } I_{low} \leq \|\mathbf{x}_t\| \leq I_{hi} \\ \textbf{false} & \text{otherwise.} \end{cases} \tag{2.3}$$

2.2.4 Foreground Detection

Subtracting the current image from the background model is straightforward. Unlike MOG or [9] which compute probabilities using costly floating point operations, our method does not involve probability calculation. Indeed, the probability estimate in [9] is dominated by the nearby training samples. We simply compute the distance of the sample from the nearest cluster mean. This is very fast and shows little difference in detection compared with the probability estimate. The subtraction operation $BGS(\mathbf{x})$ for an incoming pixel value $\mathbf{x}$ in the test set is defined as:

Algorithm for Background Subtraction

I. $\mathbf{x} = (R, G, B), \quad I \leftarrow \sqrt{R^2 + G^2 + B^2}$

II. For all codewords in $\mathcal{M}$ in Eq.2.1, find the codeword $\mathbf{c}_m$ matching to $\mathbf{x}$ based on two conditions:

- $colordist(\mathbf{x}, \mathbf{c}_m) \leq \epsilon_2$
- $brightness(I, \langle \check{I}_m, \hat{I}_m \rangle) = \textbf{true}$

Update the matched codeword as in Step II-iv in the algorithm of codebook construction.

III. $BGS(\mathbf{x}) = \begin{cases} \textbf{foreground} & \text{if there is no match} \\ \textbf{background} & \text{otherwise.} \end{cases}$

ϵ_2 is the detection threshold. The pixel is detected as foreground if no acceptable matching codeword exists. Otherwise it is classified as background.

	MOG [3]	**Kernel** [9]	**CB** (proposed)
model representation	mixture of Gaussians	kernel density	codebook
model evaluation	probability density estimation	probability density estimation	distance
parametric modeling	Yes	No	No
color metric	RGB only	normalized color r, g and s(brightness)	rescaled RGB and brightness
background memorization capacity	as much as K Gaussians hold	short-term (N samples) long-term (N samples)	almost practically infinite memory
memory usage	small	large	compact
processing speed	slow	slow	fast
model maintenance	online updating with K Gaussians	short- and long-term models	layered modeling and detection using *cache*

Table 2.1: Characteristics of background modeling algorithms

2.2.5 Review of multimode modeling techniques

Here, we compare our method with other multimode background modeling techniques - MOG [3] and Kernel [9]. The characteristics of each algorithm are listed in Table.2.1.

- Unlike MOG, we do not assume that backgrounds are multimode Gaussians. If this assumption, by chance, were correct, then MOG would get accurate parameters, and would be very accurate. But this is not always true. The background distribution could be very different from normal, as we see in compressed videos such as MPEG.

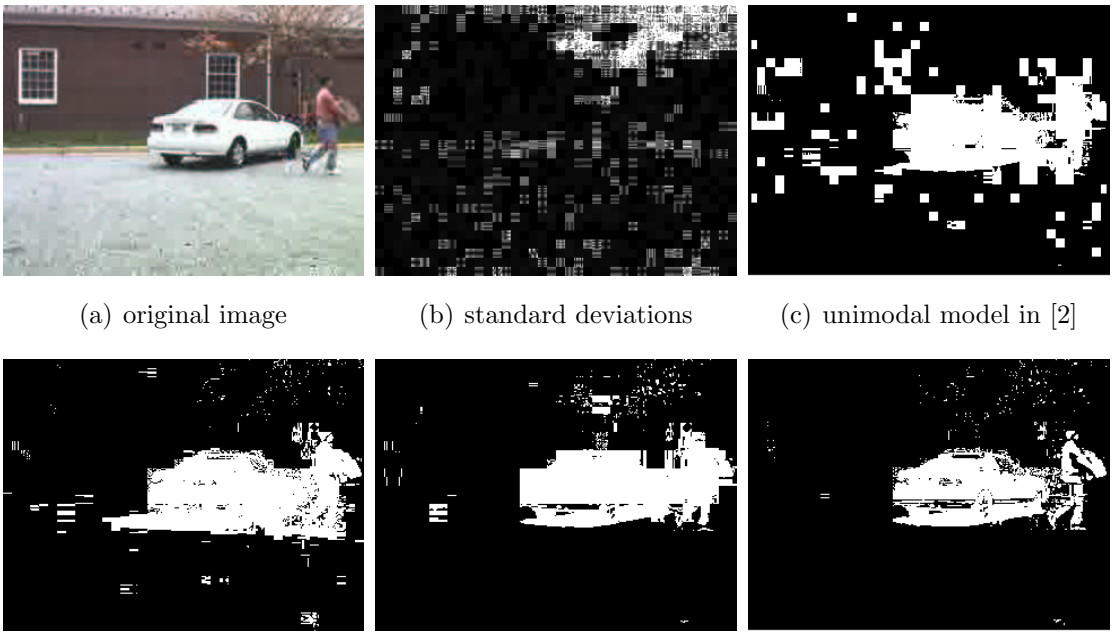
- Also, in contrast to Kernel, we do not store raw samples to maintain the background model. These samples are huge, but do not cover a long period of time. The codebook models are so compact that we can maintain them with very limited memory.

- Ours handles *multi-backgrounds* well. There is no restriction of the number of backgrounds. It can model trees which move longer than the raw sample size of Kernel. Even the rare background events, which meet the quasi-periodicity condition, survive as backgrounds.

- *Unconstrained* training using MNRL filtering allows moving foreground objects in the training sequence.

- Our codebook method does not evaluate probabilities, which is very computationally expensive. We just calculate the distance from the cluster means. That makes the operations fast.

- MOG uses the original RGB variables and doesn't separately model brightness and color. MOG currently does not model covariances, which are often large and caused by variation in brightness. It is probably best to explicitly model brightness. Kernel uses normalized colors and brightness; the normalized color has uncertainty related to brightness. To cope with the problem of illumination changes such as shading and highlights, we calculates a brightness difference as well as a color difference of rescaled RGB values.



(a) original image (b) standard deviations (c) unimodal model in [2]

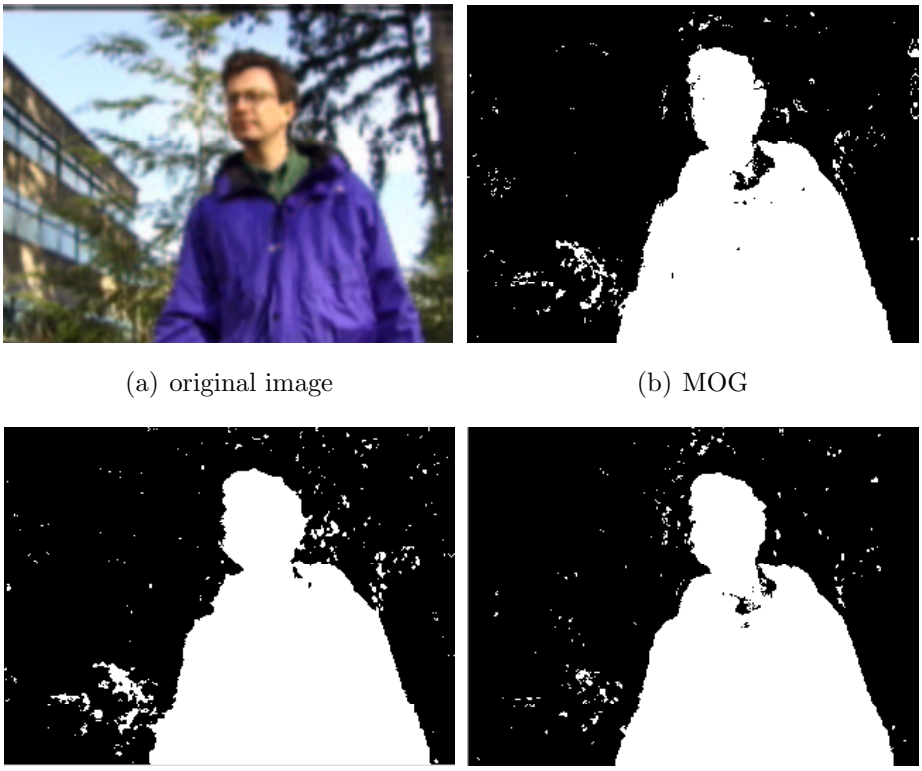
(d) MOG (e) Kernel (f) CB (proposed)

Figure 2.4: Detection results on a compressed video

2.3 Detection Results and Comparison

Most existing background subtraction algorithms fail to work with low-bandwidth compressed videos mainly due to spatial block compression that causes block artifacts, and temporal block compression that causes abnormal distribution of encoding (random spikes). Fig.2.4(a) is an image extracted from an MPEG video encoded at 70 kbits/sec. Fig.2.4(b) depicts 20-times scaled image of the standard deviations of blue(G)-channel values in the training set. It is easy to see that the distribution of pixel values has been affected by the blocking effects of MPEG. The unimodal model in Fig.2.4(c) suffers from these effects. For the compressed video, CB eliminates most compression artifacts - see Fig.2.4(c)-(f).

In a compressed video, pixel intensities are usually quantized into a few discontinuous values based on an encoding scheme. Their histograms show several



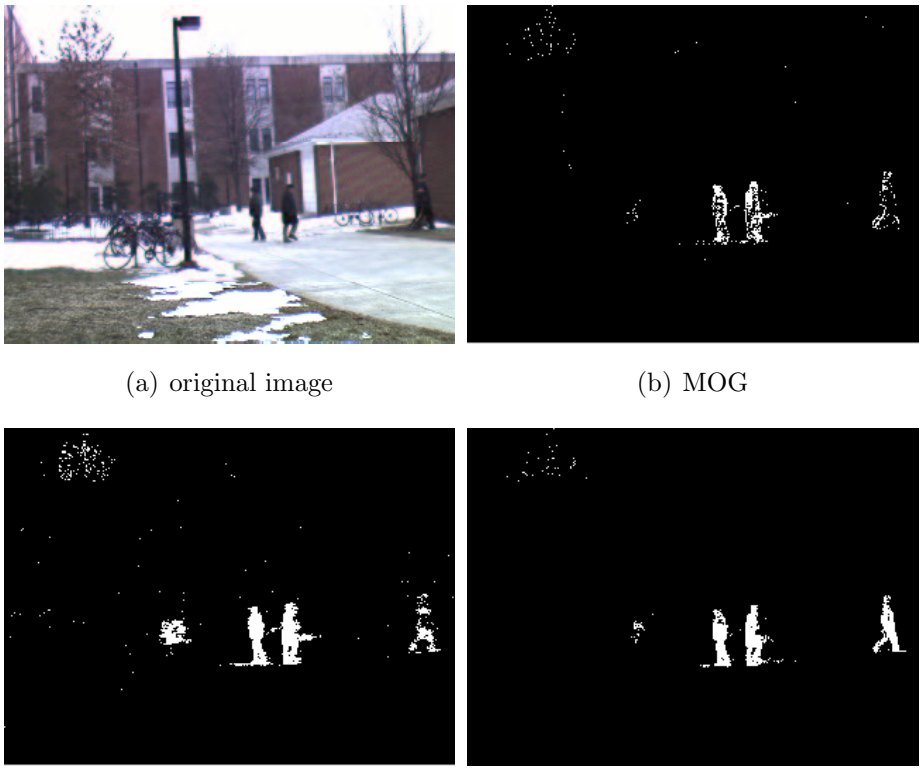
(a) original image (b) MOG

(c) Kernel (d) CB (proposed)

Figure 2.5: Detection results on multiple moving backgrounds

spike distributions in contrast to continuous bell-shaped distributions for an uncompressed video. MOG has low sensitivity around its Gaussian tails and less frequent events produce low probability with high variance. Kernel's background model, which contains a recent N-frame history of pixel values, may not cover some background events which were quantized before the N frames. If Gaussian kernels are used, the same problems occur as in the MOG case. CB is based on a vector quantization technique. It can handle these discrete quantized samples, once they survive temporal filtering (λ-filtering).

Fig.2.5 illustrates the ability of the codebooks to model multiple moving backgrounds - The trees behind the person moving significantly in the video. For the



(a) original image (b) MOG

(c) Kernel (d) CB (proposed)

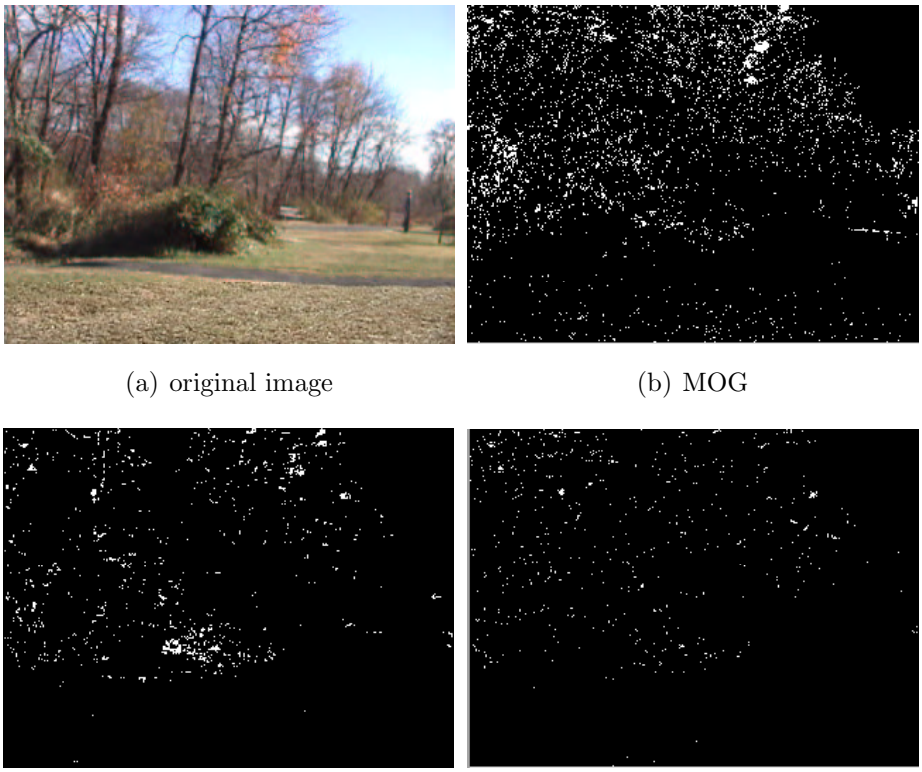
Figure 2.6: Detections results on training of non-clean backgrounds

test sequence[5] used in Fig.2.5(a), further comparison of our method was done with 10 different algorithms, and the results are described in [10].

In areas such as building gates, highways, or pathways where people walk, it is difficult to obtain good background models without filtering out the effects of foreground objects. We applied the algorithms to a test video in which people are always moving in and out a building (see Fig.2.6). By λ-filtering, our method was able to obtain the most complete background model.

Multiple backgrounds moving over a long period of time cannot be well trained with techniques having limited memory constraints. A sequence of 1000 frames

[5]We would like to thank K. Toyama and J. Krumm at Microsoft Research, for providing us with this image sequence



(a) original image (b) MOG

(c) Kernel (d) CB (proposed)

Figure 2.7: Detections results on very long-time backgrounds

	MOG	**Kernel**	**CB**
background training	8.3	40.8	39.2
background subtraction	12.1	11.1	30.7

Table 2.2: Processing speed in frames per second

recorded at 30 frames per second (fps) was trained. It contains trees moving irregularly over that period. The number of Gaussians allowed for MOG was 10. A sample of size 300 was used to represent the background. Fig.2.7 shows that CB captures most multiple background events; here we show typical false alarms for a frame containing no foreground objects. This is due to a compact background model represented by quantized codewords.

The implementation of the approach is quite straightforward and is faster than MOG and Kernel. Table 2.2 shows the speeds to process the results in Fig.2.7(b)-(d) on a 2 GHz Dual Pentium system. Note that the training time of Kernel is mostly used for reading and storing samples.

Regarding memory usage for the results in Fig.2.7(b)-(d), MOG requires 5 floating point numbers[6] RGB means, a variance, a weight for each distribution - 10 Gaussians correspond to *200* bytes. Kernel needs 3 bytes for each sample - 300 samples amount to *900* bytes. In CB, we have 5 floating point numbers $(\bar{R}, \bar{G}, \bar{B}, \check{I}, \hat{I})$ and 4 integers (f, λ, p, q) - the average[7] number of codewords in each pixel, 4 codewords, can be stored in *112* bytes.

[6]floating point: 4 bytes, integer: 2 bytes

[7]The number of codewords depends on the variation of pixel values

2.4 Automatic Parameter Estimation - ϵ_1 and ϵ_2

Automatic parameter selection is an important goal for visual surveillance systems as addressed in [40]. Two of our parameters, ϵ_1 and ϵ_2, are automatically determined. Their values depend on variation within a single background distribution, and are closely related to false alarm rates. First, we find a robust measure of background variation computed over a sequence of frames (of at least 90 consecutive frames, about 3 seconds of video data). In order to obtain this robust measure, we calculate the median color consecutive-frame difference over pixels. Then we calculate Θ (median color frame difference) which is the median over time of these median differences over space. For example, suppose we have a sequence of N images. We consider the first pair of frames, and calculate the color difference at each pixel, and take the median over space. We do this for all $N-1$ consecutive pairs, until we have $N-1$ medians. Then, Θ is the median of the $N-1$ values. In fact, an over-space median of medians over time is almost the same as Θ, while Θ is much easier to calculate with limited memory. Θ will be proportional to the within class variance of a single background. In addition, it will be a robust estimate, which is insensitive to the presence of relatively small areas of moving foreground objects. The color difference used here is defined in Eq.2.2.

Finally, we multiply a constant k by this measure to obtain $\epsilon_1 (= k\Theta)$. The default value of k is 4.5 which corresponds approximately to a false alarm rate of detection between .0001 - .002. ϵ_2 can be set to $k'\Theta$, where $(k-1) < k' < (k+1)$ but usually $k' = k$. Experiments on many videos show that these automatically

chosen threshold parameters ϵ_1 and ϵ_2 are sufficient. However, they are not always acceptable, especially for highly compressed videos where we cannot always measure the robust median accurately.

2.5 Layered modeling and detection - Model maintenance

2.5.1 Overview

Many background modeling and target detection literatures have focused on how well they model the underlying distributions of backgrounds or target foregrounds. They used the techniques such as a mixture of Gaussians [3], kernel density estimation [9, 11], high(region)-level analysis [10], color and gradient cues [6], depth measurements [5], Kalman filter [14], hidden markov model [21], markov random field [12], multiple views [22], combination with tracking [7], and so on. Many techniques tried to solve the challenging surveillance problems, for example, dynamic scenes [20, 14], crowded scene [24, 12], rain [23], underwater [27], illumination changes [28], beyond-visible-spectrum [29], non-stationary camera [25, 26], etc.

However, most background modeling techniques do not explicitly handle dynamic changes of backgrounds during detection, e.g., parked cars, left packages, displaced chairs. Even though they adapt to the changes in one way or another, they only forget the old backgrounds gradually and absorb the new background changes into the background model. Here, the meaning or importance of those background changes is ignored. Moreover, those changes are accommodated only within the capacity of the background model, i.e., the number of Gaussians in a mixture or

the number of past samples in kernel density estimation. Hence, it is desirable, in the sense of intelligent visual surveillance, to have those background changes as short-term background layers, not just a binary output of forground/background.

Research on layers for motion segmentation, object tracking, or occlusion analysis can be found in [30, 31, 32, 33, 34, 35, 36]. [30] worked on motion and appearance in layers, [31] on subspace approach, [32] on Bayesian approach, [33] on depth ordering by tracking edges, [34] on transparency manifolds, [35] on depth layers from occlusions, [36] on a background layer model. [36] is most similar to ours in that the 'background' layers are handled. However, we are interested in static layers rather than motion layers which most previous methods have considered.

The motivation of layered modeling and detection is to still be able to detect foreground objects against new backgrounds which were obtained during the detection phase. If we do not have those short-term background layers, interesting foreground objects (e.g., people) will be detected mixed with other stationary objects (e.g., cars). The short-term backgrounds can be labelled with the time when they first appeared static so that they can be represented in temporal order.

The background layers are embedded into the existing technique using a codebook model. Please note that it is a pixel-based approach which makes layers defined initially on a per-pixel basis.

An overview of the layered modeling algorithm is given in Sec.2.5.2. Sec.2.5.3 discusses several important questions which need to be considered when constructing layered models. Experimental results showing surveillance examples are shown in Sec.2.5.4.

2.5.2 Model updating for background changes

As noted in Sec.2.5.1, the scene can change after initial training, for example, by parked cars, displaced books, etc. These changes should be used to update the background model. We achieve this by defining an additional model $\mathcal{H}$ called a *cache* and three parameters described below:

- $T_{\mathcal{H}}$: the threshold for MNRL of the codewords in $\mathcal{H}$;
- T_{add}: the minimum time period required for addition, during which the codeword must reappear;
- T_{delete}: a codeword is deleted if it has not been accessed for a period of this long.

The periodicity of an incoming pixel value is filtered by $T_{\mathcal{H}}$, as we did in the background modeling (Sec.2.2.2). The values re-appearing for a certain amount of time (T_{add}) are added to the background model as short-term background. Some parts of a scene may remain in the foreground unnecessarily long if adaptation is slow, but other parts will disappear too rapidly into the background if adaptation if fast. Neither approach is inherently better than the other. The choice of this adaptation speed is problem dependent.

We assume that the background obtained during the initial background modeling is long-term. This assumption is not necessarily true, e.g., a chair can be moved after the initial training, but, in general, most long-term backgrounds are obtainable during training. Background values not accessed for a long time (T_{delete}) are deleted from the background model. Optimally, the long-term codewords are augmented with permanent flags indicating they are not to be deleted*. The permanent flags can be applied otherwise depending on specific application needs.

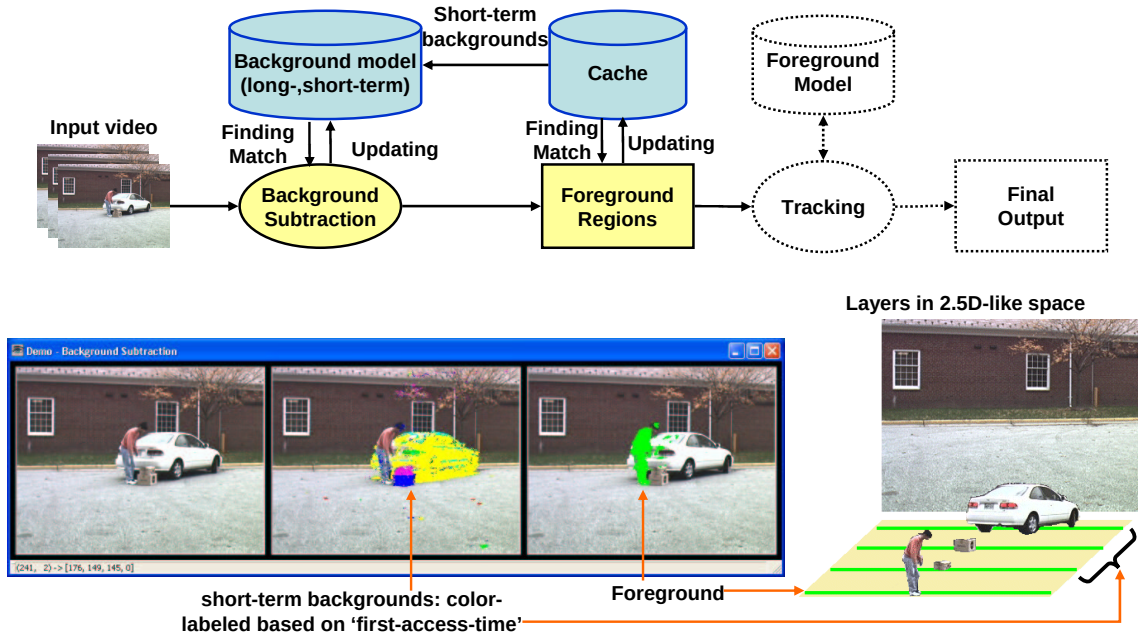


Figure 2.8: The overview of our approach with short-term background layers: the foreground and the short-term backgrounds can be interpreted in a different temporal order. The diagram items in dotted line, such as Tracking, are added to complete a video surveillance system.

Thus, a pixel can be classified into four subclasses - (1) background found in the long-term background model, (2) background found in the short-term background model, (3) foreground found in the cache, and (4) foreground not found in any of them. The overview of the approach is illustrated in Fig.2.8. This adaptive modeling capability allows us to capture changes to the background scene. The detailed procedure is given below.

Algorithm for Background Update

I. After training, the background model $\mathcal{M}$ is obtained as in Eq.2.1. Create a new model $\mathcal{H}$ as a cache.

II. For an incoming pixel $\mathbf{x}$, find a matching codeword in $\mathcal{M}$. If found, update the codeword.

III. Otherwise, try to find a matching codeword in $\mathcal{H}$ and update it. For no match, create a new codeword $\mathbf{h}$ and add it to $\mathcal{H}$.

IV. Filter out the cache codewords which do not occur quasi-periodically (That is, their λ's are larger than the threshold $T_{\mathcal{H}}$).

$$\mathcal{H} \leftarrow \mathcal{H} - \{\mathbf{h}_i | \mathbf{h}_i \in \mathcal{H},\ \ \lambda(\mathbf{h}_i) > T_{\mathcal{H}}\}$$

V. Among the cache codewords which survive from the filtering in Step IV, move the ones, staying enough time in $\mathcal{H}$ to be determined as short-term backgrounds, to $\mathcal{M}$ (Their *first* access times are larger than T_{add}).

$$\mathcal{M} \leftarrow \mathcal{M} \cup \{\mathbf{h}_i | \mathbf{h}_i \in \mathcal{H},\ \ p(\mathbf{h}_i) > T_{add}\}$$

VI. Delete the codewords not accessed for a long time from $\mathcal{M}$ (Their *last* access times are larger than T_{delete}). But do not delete a codeword augmented with a permanent flag.

$$\mathcal{M} \leftarrow \mathcal{M} - \{\mathbf{c}_i | \mathbf{c}_i \in \mathcal{M},\ \ q(\mathbf{c}_i) > T_{delete},\ \ permanent(\mathbf{c}_i) = \text{no}^*\}$$

VII. Repeat the process from the Step II.

Many short-term background layers can be formed as changes to the background occur. The parameters $T_{\mathcal{H}}$, T_{add} and T_{delete} need to be controlled based on the specific application needs or the semantics of foreground objects.

The first-access-time of a codeword, p, can be used to label its background layer. Based on this temporal information, layers can be ordered in time-depth and temporal segmentation can also be performed.

2.5.3 Issues for Background Updating

There are several related background-updating issues that need to be considered for practical visual surveillance applications. In this section, those issues are discussed along with related references and possible solutions. As noted in [10], larger systems seeking a high-level understanding of image sequences use background subtraction as a component. A background maintenance module handles

the default model for everything in a scene that is not modeled explicitly by other processing modules. Thus, the module performing background maintenance should not attempt to extract the semantics of foreground object on its own.

Spatial integration : Time-stamps of 'first-access-time' are assigned to background layers on each pixel as mentioned in the last paragraph in Sec.2.5.2. It is possible to segment the object by grouping pixels with similar time-stamps at close distance, without or with the help of 'spatial segmentation' (See [39] for segmentation techniques). However, note that a region of temporal segmentation may not correspond to a physical object, and vice versa.

Move in or out : There are two cases of background model updating - (1) A new object (blob) comes in to the scene or displaces, and then stops to be a short-term background, (2) An existing object modeled as background leaves the original place. The hole left behind would be labelled as short-term background.

The object is a connected component in a binary foreground map. Here, the object is assumed to be rigid. Over the boundary pixels, we can apply a color similarity test (or a symmetric neighbor filter [37]) to classify a move-in or move-out case as shown in Fig.2.9.

Human VS. stationary object : How to deal with a person who becomes almost stationary? There would be 'foreground aperture' or 'sleeping person' problems as addressed in [10]. Depending on the semantics of foreground objects, we may not want to let stationary people become background layers.

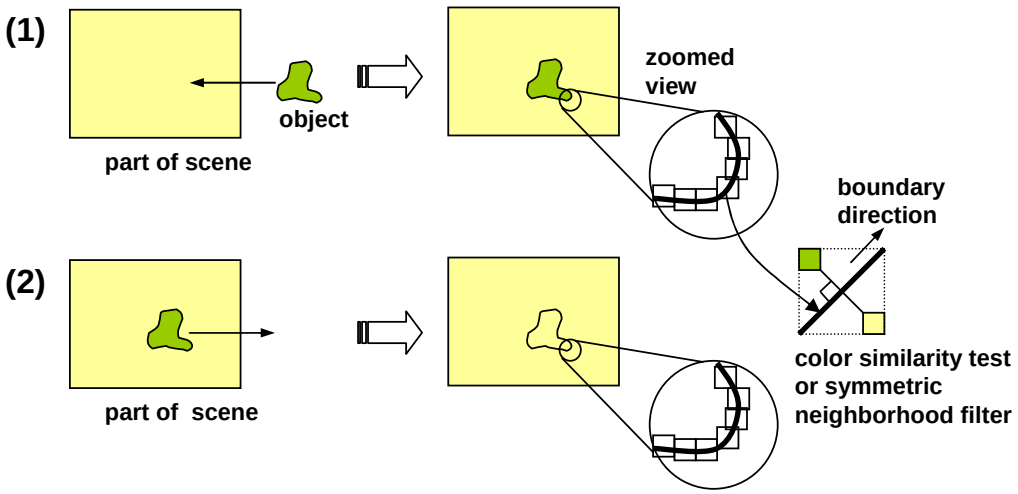


Figure 2.9: Two cases of changed backgrounds. The case (2) shows almost homogeneous neighborhoods over the boundary while different colors are observed on the opposite sides along the boundary direction in the case (1).

A higher-level module needs to provide feedback to background maintenance about what pixels should not be adapted into the background. We could determine that the tracked object is a person or a group of people beforehand by keeping a foreground model. Sometimes, the human object boundary may not be perfectly motionless. Several heuristics to identify human objects by head detection, boundary analysis or vertical histograms were proposed in [24, 38]

Pre-labelled environments : Some fixtures like doors or gates need to be labelled before performing visual surveillance tasks since those are always in one of the pre-defined states - widely open, ajar, or closed. Many surveillance scenes involve doors or gates where interesting human activity events can occur. Moreover, in most cases, opening or closing a door causes illumination changes on the surrounding areas and, as a result, detection algorithms give

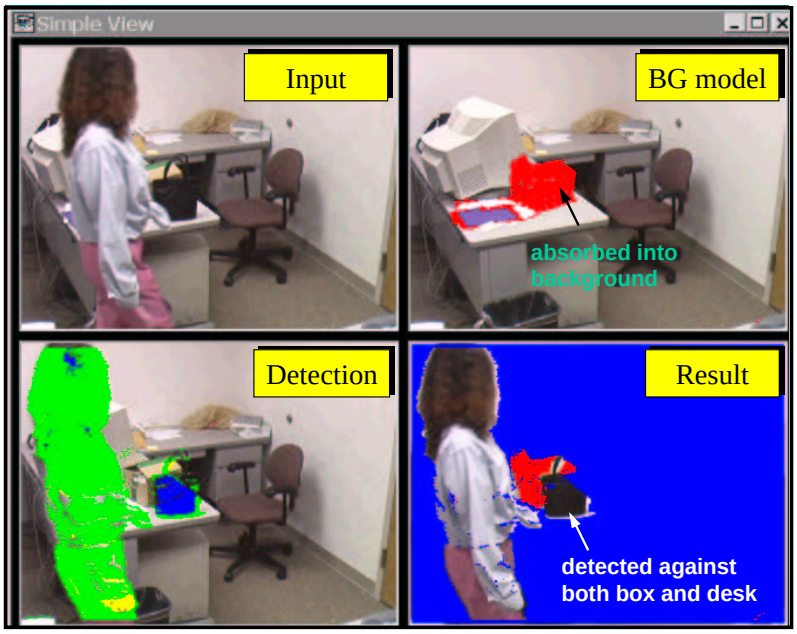


Figure 2.10: Layered modeling and detection - A woman placed a box on the desk and then the box has been absorbed into the background model as short-term. Notice that the paper on the desk was displaced by the box and then also became a short-term background layer. Then a purse is put in front of the box. The purse is detected against both the box and the desk.

false alarms.

One could manually store representative states of those areas as short-term backgrounds on the background model before performing actual detection. If that kind of pre-processing is not available or the environment is not controllable, the area needs to be specially labelled as a door or a gate, and then handled differently, i.e., detecting moving objects not by subtracting from a background model but by matching with foreground models only.

2.5.4 Experimental results - examples

Fig.2.10 shows detection of an object against both long-term backgrounds and a short-term background layer.

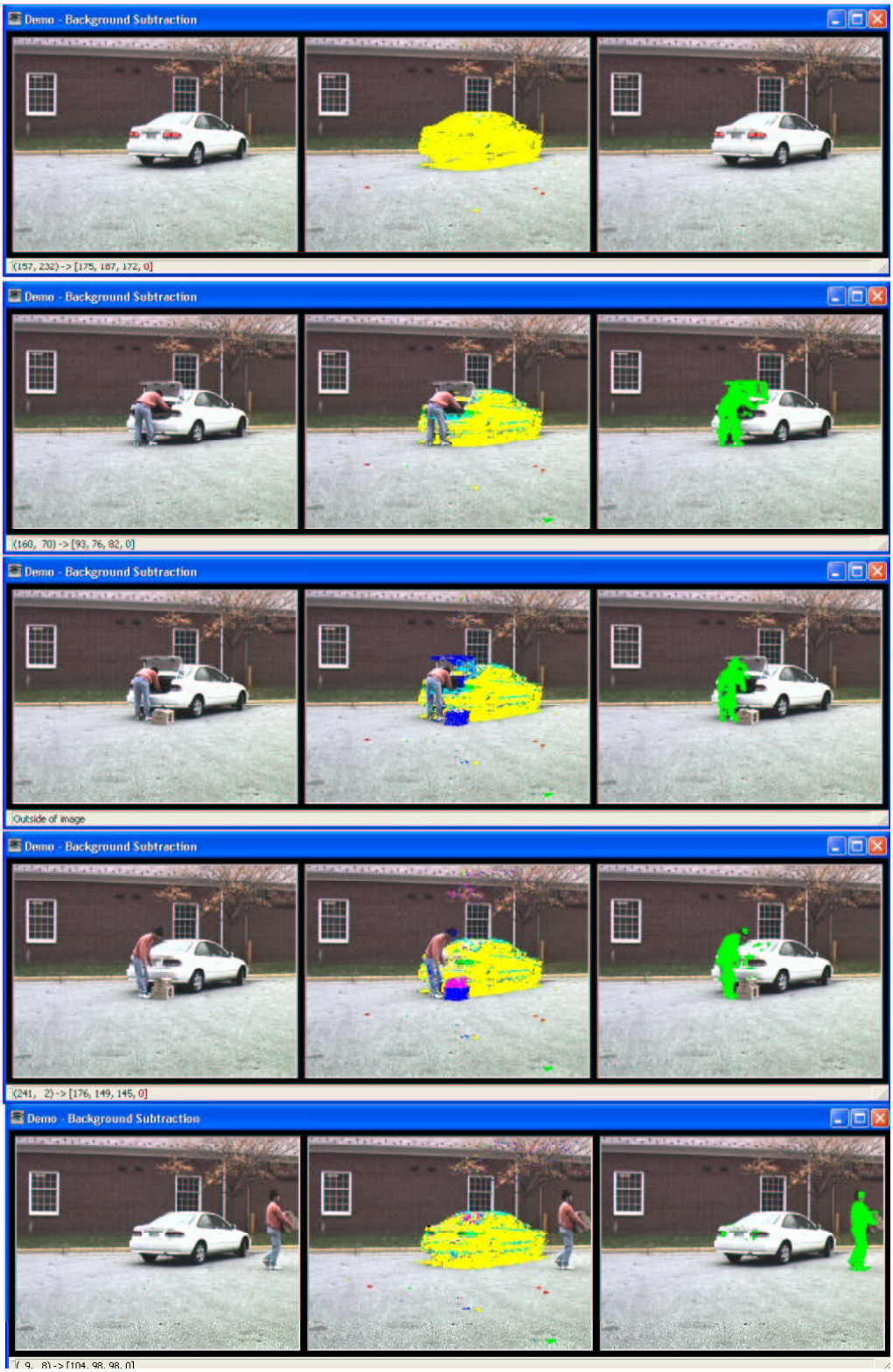
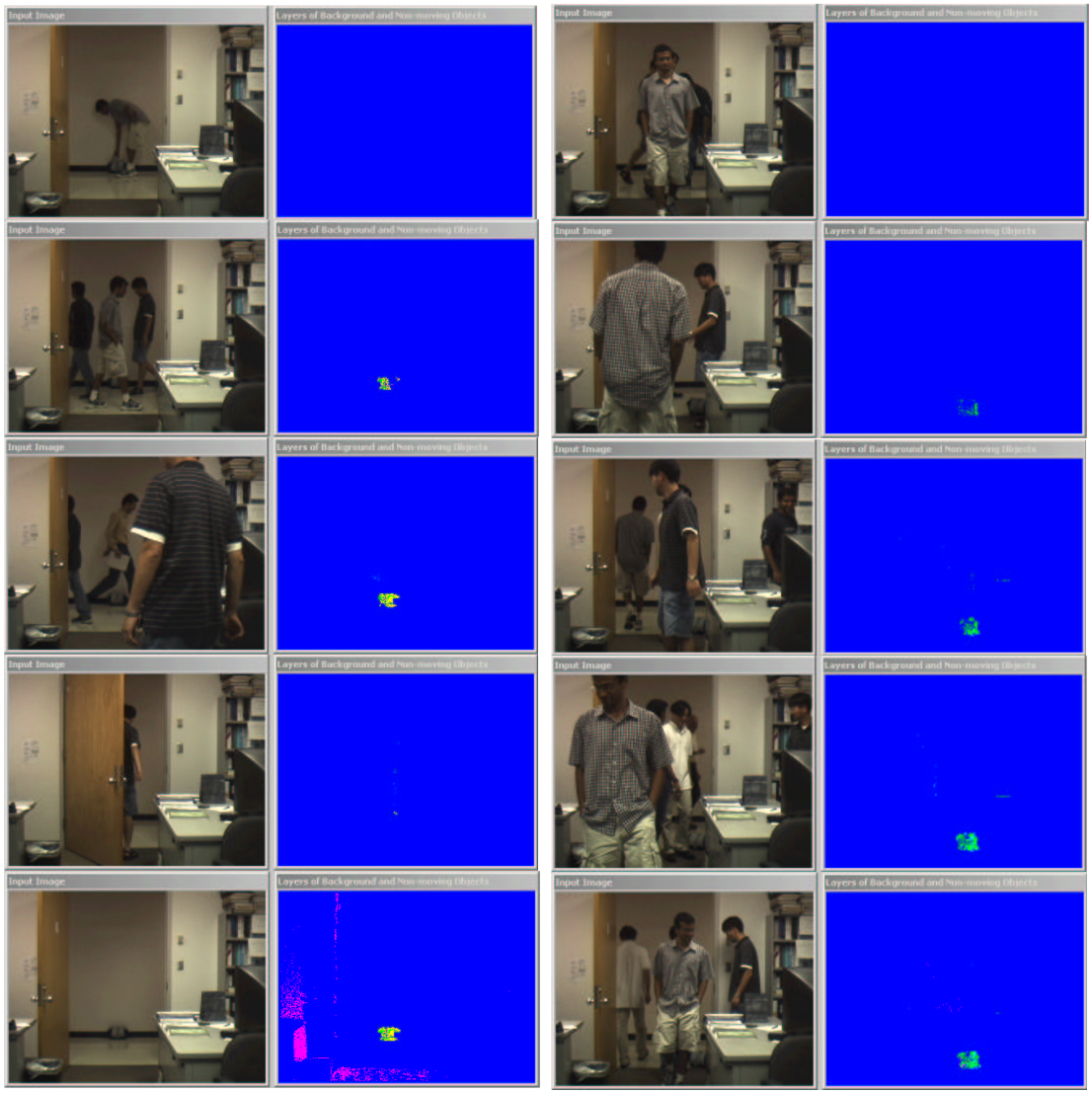


Figure 2.11: The leftmost column: original images, the middle column: color-labelled short-term backgrounds, the rightmost column: detected foreground. The video shows that a man parks his car on the lot and takes out two boxes. He walks away to deliver them.



(a) (b)

Figure 2.12: The sample frames are shown in the order of time along with short-term background layers in the second column. (a): A bag is placed by somebody and left unattended. A short-term background layer is formed. It is still memorized as a layer after the door was closed and then opened. (b): While several people walk in and out the office, a bag has been left without any attention. Even with severe occlusion by walking people, the bag stands out as a layer.

Fig.2.11 is a more interesting example which can be used for the further analysis of scene change detection. After parking a car, a man unloads two boxes one after another. The car and the two boxes are labelled with different coloring based on their 'first-access-times' as short-term backgrounds while the man is still detected as an active foreground. A car becomes a far-most background layer and then two boxes create two different layers against the car layer.

As shown in Fig.2.12(a),2.12(b), a package left unattended for a long time would be one of most demanding surveillance targets. Two such scenarios are presented here. To be precise on detection of unattended objects, a high-level analysis to identify 'unattendedness' is required along with this low-level detection.

2.6 Adaptive codebook updating - detection under global illumination changes

Global illumination changes (for example, due to moving clouds) make it difficult to conduct background subtraction in outdoor scenes. They cause over-detection, false alarms, or low sensitivity to true targets. Good detection requires equivalent false alarm rates over time and space. We discovered from experiments that variations of pixel values are different (1) at different surfaces (shiny or muddy), and (2) under different levels of illumination (dark or bright). Codewords should be adaptively updated during illumination changes. Exponential smoothing of codeword vector and variance with suitable learning rates is efficient in dealing with illumination changes. It can be done by replacing the updating formula of $\mathbf{v}_m$ with

$$\mathbf{v}_m \leftarrow \gamma \mathbf{x}_t + (1-\gamma)\mathbf{v}_m$$

and appending

$$\sigma_m^2 \leftarrow \rho\delta^2 + (1-\rho)\sigma_m^2$$

to Step II-iv of the algorithm for codebook construction. γ and ρ are learning rates. Here, σ_m^2 is the overall variance of color distortion in our color model, not the variance of RGB. σ_m is initialized when the algorithm starts. Finally the function $colordist()$ in Eq.2.2 is modified to

$$colordist(\mathbf{x}_t, \mathbf{v}_i) = \frac{\delta}{\sigma_i}.$$

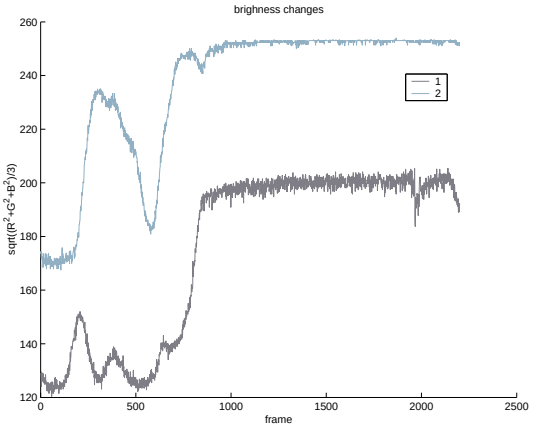
We tested a PETS'2001[8] sequence which is challenging in terms of multiple targets and significant lighting variation. Fig.2.13(a) shows two sample points (labelled 1 and 2) which are significantly affected by illumination changes and Fig.2.13(b) shows the brightness changes of those two points. As shown in Fig.2.13(d), adaptive codebook updating eliminates the false detection which occurs on the roof and road in Fig.2.13(c).

2.7 Conclusion and Discussion

Our new adaptive background subtraction algorithm, which is able to model a background from a long training sequence with limited memory, works well on moving backgrounds, illumination changes (using our color distortion measures), and compressed videos having irregular intensity distributions. It has other desirable features - unconstrained training and layered modeling/detection. Comparison with

[8]IEEE International Workshop on Performance Evaluation of Tracking and Surveillance 2001 at http://www.visualsurveillance.org/PETS2001

(a) original image - frame 1



(b) brightness changes

(c) before adaptive updating

(d) after adaptive updating

Figure 2.13: Results of adaptive codebook updating for detection under global illumination changes. Detected foregrounds on the frame 1105 are labelled with green color.

other multimode modeling algorithms shows that the codebook algorithm has good properties on several background modeling problems.

In summary, our major contributions are as follows:

1. We propose a background modeling technique efficient in both memory and speed. Experiments show that nearest neighbor 'classification', which is computationally very efficient, is as effective as probabilistic classification (both kernel and MOG) for our application. Practically, even when computing probabilities of pixel measurements coming from the background, these probabilities are dominated by the nearest component of the background mixture.

2. The most important lesson, based on our experience, for analyzing color videos is that using an appropriate color model is critical for obtaining accurate detection, especially in low light conditions such as in shadows. Using RGB directly lowers detection sensitivity because most of the variance at a pixel is due to brightness, and absorbing that variability into the individual RGB components results in a lower true detection rate for any desired false alarm rate. In other words, an algorithm would have to allow greater color variability than the data actually requires in order to accommodate the intrinsic variability in brightness. Using normalized colors, on the other hand, is undesirable because of their high variance at low brightness levels; in order to maintain sufficiently low detection error rates at low brightness, one necessarily sacrifices sensitivity at high brightness. This is due to using an angular measure between normalized color coordinates for detection. The color model proposed

here, on the other hand, maintains a constant false alarm rate across, essentially, the entire range of brightness levels. One would expect that modifying other background subtraction algorithms, such as the MOG algorithm, to use this more appropriate color model would bring their performance much closer to that of the codebook algorithm.

Automatic parameter selection is an important goal for visual surveillance systems as addressed in [40]. Two of our parameters, ϵ_1 in Section 2.2.1 and ϵ_2 in Section 2.2.4, can be automatically determined. Their values depend on variation within a single background distribution, and are closely related to false alarm rates. Preliminary experiments on many videos show that automatically chosen threshold parameters ϵ_1 and ϵ_2 are sufficient. However, they are not always acceptable, especially for highly compressed videos where we cannot always measure the robust parameter accurately. In this regards, further investigation could be done to obtain robust parameters.

Chapter 3

Performance Evaluation of Sensitive Target Detection

3.1 PDR - A performance evaluation method

3.1.1 Concept

The purpose of PDR analysis is to measure the detection sensitivity of a BGS algorithm without assuming knowledge of the actual foreground distribution. The basic idea is to measure how far apart the two distributions must be in order to achieve a certain detection rate, or stated otherwise, given a false alarm rate (FA-rate), to determine detection rate as a function of the difference of the foreground from the background. It is similar to the *Just Noticeable Difference* (JND) typically used in comparing psychophysical magnitudes.

In general, detection accuracy depends on the algorithm and its parameters, shapes of the foreground and background distributions, and how far apart they are. In ROC, we assume we are given both foreground and background data of particular distribution shape and separation. We may vary the algorithm's parameters to obtain a certain combined false alarm rate and miss detection rate (or detection rate). Whereas, in PDR, we do not need to know exactly what the distributions are. The basic assumption made is that the shape of the foreground distribution is locally similar to that of the background distribution; however, the foreground distribution

of small ("just-noticeable") contrast will be a shifted or perturbed version of the background distribution. This assumption is fairly reasonable because, in modeling video, any object with its color could be either background or foreground, e.g., a parked car could be considered as a background in some cases; in other cases, it could be considered a foreground target. Furthermore, by varying algorithm parameters we determine not a pair of error rates but a relation among the false alarm and detection rates and the distance between the distributions.

Given the parameters to achieve a certain fixed FA-rate, the analysis is performed by shifting or perturbing the entire BG distributions by vectors in uniformly random directions of RGB space with fixed magnitude Δ, computing an average detection rate as a function of contrast Δ. It amounts to simulating possible foregrounds at certain color distances. In the PDR curve, we plot the detection rate as a function of the perturbation magnitude Δ given a particular FA-rate.

3.1.2 PDR algorithm

The PDR algorithm is presented step-by-step as:

PDR Algorithm

---

1. First, we train the BGS algorithm on N training background frames, adjusting parameters as best we can to achieve a target FA-rate[1] [2]. It is averaged over the training frames. The FA-rate would be practical in processing the video. Typically this will range from .01% to 1% depending on video image quality.

[1]Note that there are more than one combination of parameter settings that might produce the target FA-rate. In our experiments, we varied one or two major parameters directly affecting the detection performance with other minor parameters fixed. The best combination of parameters of the target FA-rate was chosen by visual inspection of real-object detection.

[2]Obtaining the FA-rate could be done by "leave all in" or "cross validation" test protocols [42]

2. To obtain a test foreground at color contrast Δ, we pass through the N background frames again. For each frame, we perturb a random sample of M pixel values (R_i, G_i, B_i) by a magnitude Δ in uniformly random directions. The perturbed, foreground color vectors (R', G', B') are obtained by generating points randomly distributed on the color sphere with radius Δ.

3. Then we test the BGS algorithms on these perturbed, foreground pixels and compute the detection rate for the Δ.

4. By varying the foreground contrast Δ, we obtain a monotone increasing PDR graph of detection rates.

In some cases, one algorithm will have a graph which dominates that of another algorithm for all Δ. In other cases, one algorithm may be more sensitive only in some ranges of Δ. Most algorithms perform very well for a large contrast Δ, so we are often concerned with small contrasts ($\Delta < 40$) where differences in detection rates may be large.

3.2 Results

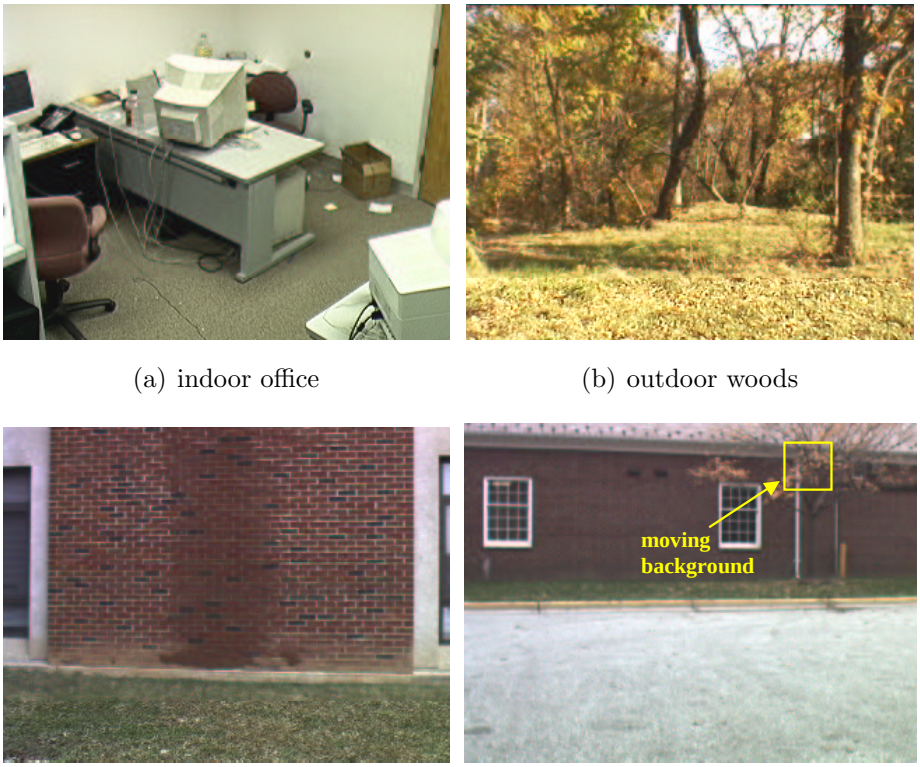
3.2.1 Tested algorithms and experiment setups

In this study, we compare four algorithms shown in Table 3.1. Since the algorithm in [9] accepts normalized colors (KER) or RGB colors (KER.RGB) as inputs, it has two separate graphs. Figure 3.1 shows representative empty images from four test videos. Note that the MOG implementation here is based on the original algorithm in [3], which uses RGB colors directly.

To generate PDR curves, we collected 100 empty consecutive frames from each video. 1000 points are randomly selected at each frame. That is, for each Δ, $(100)\times(1000)$ perturbations and detection tests were performed. Those 100

Name	Background subtraction algorithm
CB	codebook-based method described in [62]
MOG	mixture of Gaussians described in [3]
KER and **KER.RGB**	non-parametric method using kernels described in [9]
UNI	unimodal background modeling described in [2]

Table 3.1: Four algorithms used in performance evaluation



(a) indoor office (b) outdoor woods

(c) red-brick wall (d) parking lot

Figure 3.1: The sample empty-frames of four videos used in the experiments

empty frames are also used for training background models. During testing, no updating of the background model is allowed. For the non-parametric model in KER and KER.RGB, a sample size 50 was used to represent the background. The maximum number of Gaussians allowed in MOG is 4 for the video having stationary backgrounds and 10 for moving backgrounds. We do not use a single FA-rate for all four videos. The FA-rate for each video is determined by three factors - video quality, whether it is indoor or outdoor, and good real foreground detection results for most algorithms. The FA-rate chosen this way is practically useful for each video. The threshold value for each algorithm has been set to produce a given FA-rate. In the case of MOG, the learning rate, α, was fixed to 0.01 and the minimum portion of the data for the background, T, was adjusted to give the desired FA-rate. Also, the cluster match test statistic was set to 2 standard deviations. Unless noted otherwise, the above settings are used for the PDR analysis.

3.2.2 Indoor and outdoor videos

Figures 3.2 and 3.3 show the PDR graphs for the indoor and outdoor videos in Figures 3.1(a) and 3.1(b) respectively.

For the indoor office video, consisting almost entirely of stationary backgrounds, CB and UNI perform better than the others. UNI, designed for unimodal backgrounds, has good sensitivity as expected. KER performs intermediately. MOG and KER.RGB do not perform as well for small contrast foreground Δ, probably because, unlike the other algorithms, they use original RGB variables and don't separately model brightness and color. MOG currently does not model covariances

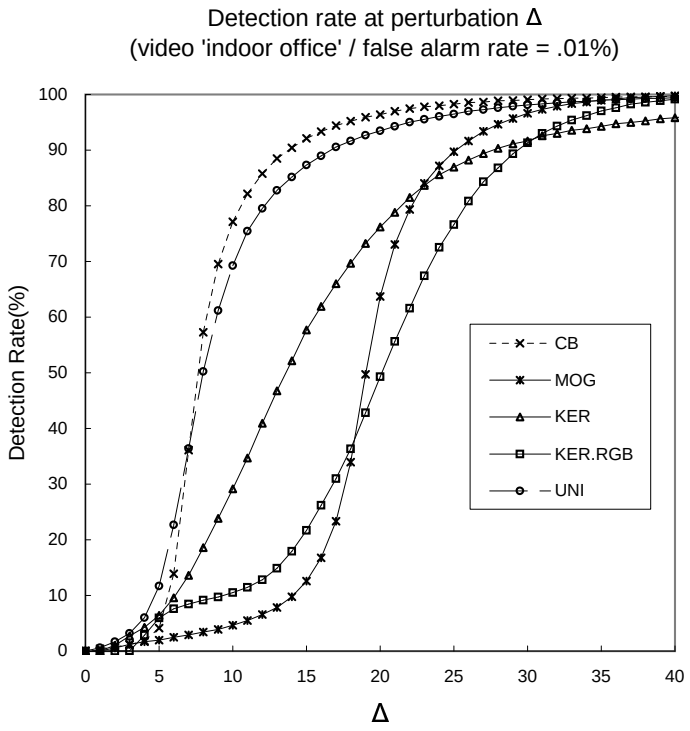


Figure 3.2: PDR for 'indoor office' video in Figure 3.1(a)

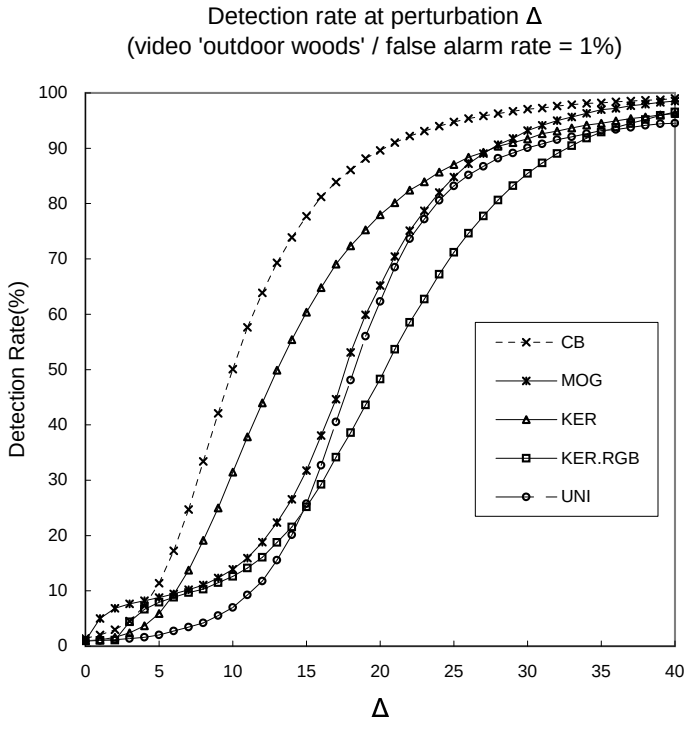


Figure 3.3: PDR for 'outdoor woods' video in Figure 3.1(b)

(a) a 'red-brick wall' frame including a person in a red sweater

(b) detection using MOG

(c) detection using CB

Figure 3.4: Sensitive detection at small delta

which are often large and caused by variation in brightness. It is probably best to explicitly model brightness. MOG's sensitivity is consistently poor in all our test videos, probably for this reason, and not due to the density representation as Gaussian mixtures.

For the outdoor video, all algorithms perform somewhat worse even though the FA-rate has been increased to 1% from .01%. CB and KER, both of which model mixed backgrounds and separate color/brightness, are most sensitive, while, as expected, UNI does not perform well as in the indoor case. KER.RGB and MOG are also less sensitive outdoors, as before indoors.

3.2.3 Detection sensitivity - a real example

Figure 3.4 depicts an example of foreground detection, showing real differences in detection sensitivity for two algorithms. These differences reflect the performance predicted by the PDR graph in Figure 3.5. The video image in Figure 3.4(a) shows someone with a red sweater standing in front of a brick wall of somewhat different reddish color. There are detection holes through the sweater (and face) in the MOG

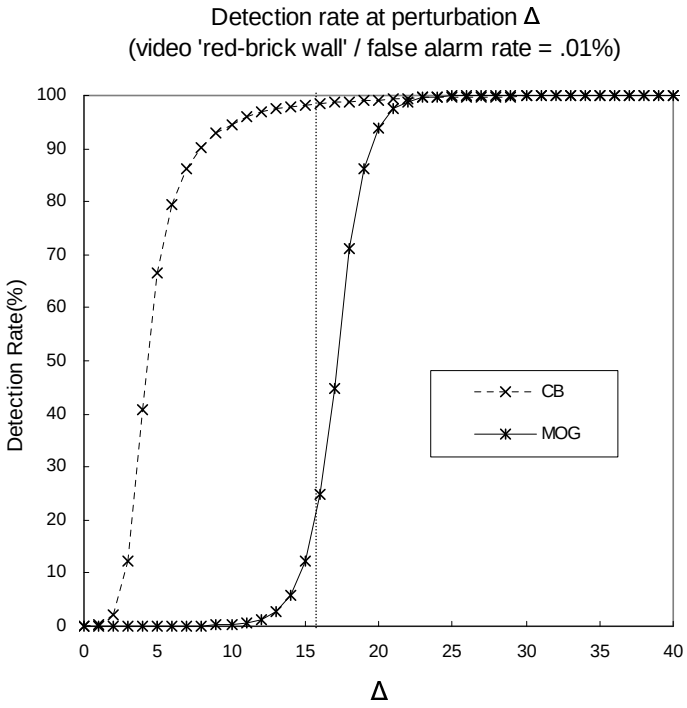


Figure 3.5: PDR for 'red-brick wall' video in Figure 3.1(c)

result (Figure 3.4(b)) . The CB result in Figure 3.4(c) is much better at this small contrast. After inspection of the image, the magnitude of contrast Δ was determined to be about 16 in missing spots. This was due to difference in color balance and not overall brightness. Figure 3.5 shows a large difference in detection for this contrast, as indicated by the vertical line.

3.2.4 Multiple moving backgrounds

Figures 3.6 shows how sensitively the algorithms detect foregrounds against a scene containing moving backgrounds (trees). In order to sample enough moving background events, 300 frames are allowed for training. A window is placed to represent 'moving backgrounds' as shown in Figure 3.1(d). PDR analysis is performed on the window with the FA-rate obtained only within the window - a 'window' false

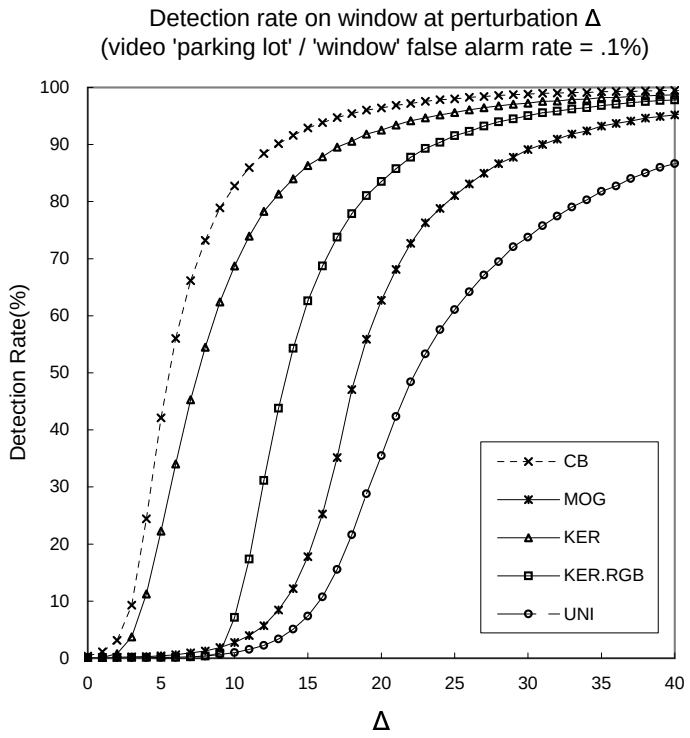


Figure 3.6: PDR for window on moving background (Figure 3.1(d))

alarm rate (instead of 'frame' false alarm rate).

The PDR graph (Figure 3.6) for the moving background window is generally shifted right, indicating reduced sensitivity of all algorithms for moving backgrounds. Also, it shows differences in performance among algorithms, with CB and KER performing best. These results are qualitatively similar to those for the earlier example of outdoor video shown in Figure 3.3. We can offer the same explanation as before: CB and KER were designed to handle mixed backgrounds, and they separately model brightness and color. In this video experiment, we had to increase the background sample size of KER (also that of KER.RGB) to 270 frames from 50 in order to achieve the target FA-rate in the case of the moving background window. It should be noted that CB, like MOG, usually models background events over a

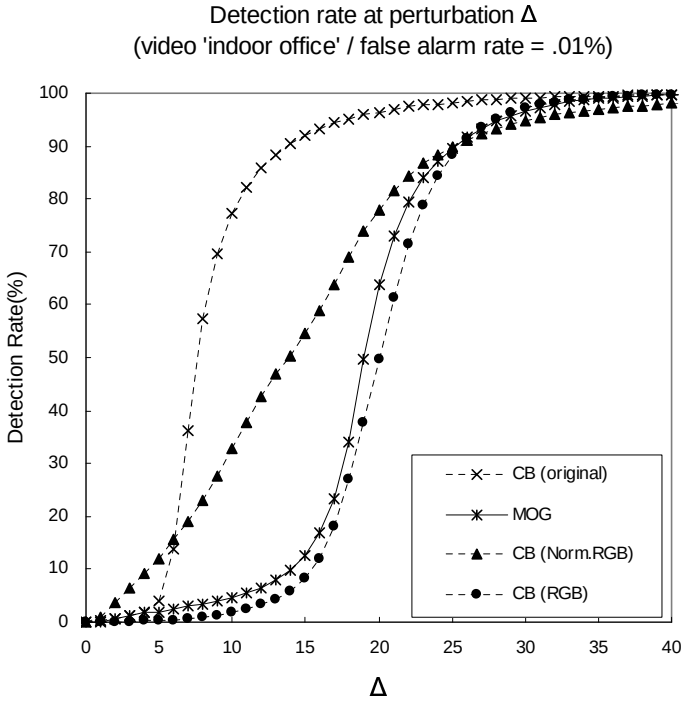


Figure 3.7: PDR of different color models for the video in Figure 3.1(a)

longer period than KER.

3.2.5 Different color models

The color model being used could affect the detection performance. We tweaked the original CB method into two new CB versions having the color models of 'RGB' and 'normalized RGB'. The RGB version just takes the absolute RGB difference between a pixel and a background model for its color distortion measure. Likewise, the Norm.RGB version uses the normalized color difference. Figure 3.7 shows that the performance of the tweaked CB versions on the video in Figure 3.1(a) are degraded, and for the RGB color model are almost indistinguishable from MOG as expected. The graph of MOG has been added for reference.

This leads us to conclude that an important lesson for analyzing color videos

is that using an appropriate color model is more critical for obtaining accurate detection, especially in low light conditions such as in shadows than the density representation employed. Using RGB directly lowers detection sensitivity because most of the variance at a pixel is due to brightness, and absorbing that variability into the individual RGB components results in a lower true detection rate for any desired false alarm rate. In other words, an algorithm would have to allow greater color variability than the data actually requires in order to accommodate the intrinsic variability in brightness. As noted in [17, 62], using normalized colors, on the other hand, is undesirable because of their high variance at low brightness levels; in order to maintain sufficiently low detection error rates at low brightness, one necessarily sacrifices sensitivity at high brightness. This is due to using an angular measure between normalized color coordinates for detection. The color model proposed in the original CB algorithm, on the other hand, maintains a constant false alarm rate across, essentially, the entire range of brightness levels. The CB method calculates a brightness difference (a ratio of RGB absolute values) and a color difference which rescales codeword RGB values to the brightness of the current, tested pixel.

3.3 Conclusions and future work

3.3.1 Conclusions

We presented a perturbation method for measuring sensitivity of BGS algorithms. The PDR method does not require foreground targets in videos or knowledge of actual foreground distributions. PDR analysis does not consider all possible background or foreground distributions; it considers only those relevant to one video,

scene and camera. It assumes that the foreground, when it has small contrast to the background locally, has a distribution similar in form to the background, but shifted or perturbed.

PDR analysis has two advantages over the commonly used ROC analysis: (1) It does not depend on knowing foreground distributions, (2) It does not need the presence of foreground targets in the video in order to perform the analysis, while this is required in the ROC analysis. Because of these considerations, PDR analysis provides practical general information about the sensitivity of algorithms applied to a given video scene over a range of parameters and FA-rates. In ROC curves, we obtain one detection rate for a particular FA-rate for a particular foreground and contrast.

We have applied the PDR analysis to four various BGS algorithms and four videos of different types of scenes. The results seem to be understandable, reflecting obvious differences among the algorithms as applied to the particular type of background scenes. We also provided a real video example of differences among the algorithms with respect to sensitive foreground detection which appears consistent with the PDR simulation.

There are limitations. The method doesn't model motion blur of moving foreground objects. Also in the case of mixed (moving) backgrounds, the simulated foreground distributions will be mixed (as plants or flags moving in the foreground); usually, though, foreground targets are from unimodal distributions. It should be noted, however, that the overall detection rates will be nearly the same if the clusters of the mixed distributions are well separated (compared to the usual small contrast

delta). An important limitation is that foreground objects often will have shading and reflection effects on backgrounds, and these are ignored although they are important for choosing a proper, practical false alarm rate for real video analysis. (We have chosen practical false alarm rates for the videos used in this study.)

PDR does not predict the overall performance of a background subtraction algorithm, but shows detection rates for possible foreground targets given the background scene. ROC analysis is also very useful if a specific real target is known and crucial to the application. We would not generally claim that one algorithm is better than another just from PDR analysis. There are other important performance criteria which are not compared, such as processing speed, memory capacity, online model update, etc..

3.3.2 Future Work

The present method would seem to be useful for qualitative comparison of sensitivity of different algorithms, as well as comparison of choice of parameters for a particular algorithm with respect to sensitivity. In the future, the present method could be extended to measure local detection rates throughout the frame of the scene or varying over time. This might have application to localized parameter estimation, e.g. of detection/adaptation parameters in different parts of the frame of the scene.

In the parameter space[3], every combination of parameters determines its FA-

[3]The dimension of this space is determined by the number of parameters. A small number of major parameters directly affecting the detection performance (with other minor parameters fixed) reduces the size of the parameter space to be explored

rate. One could select those combinations that produce the target FA-rate, then plot a family of PDR graphs for them. One could then choose the algorithm parameters that provide best detection sensitivity with respect to the PDR analysis.

Chapter 4

Multi-view Multi-target Multi-Hypothesis Segmentation and Tracking of People

This chapter is organized as follows. Sec.4.1 presents a human appearance model. A framework for segmenting and tracking occluded people moving on a ground plane is presented in Sec.4.2. This is based on a method to integrate practical information from segmented blobs across views on a top-view reconstruction, based on knowledge of each camera's ground plane homography. In Sec.4.3, the multi-view tracker is extended to a multi-hypothesis framework (M^3Tracker) using particle filtering. We demonstrate the experimental results of the proposed approach on video sequences in Sec.. Finally, conclusions are presented in the last section.

4.1 Human appearance model

Haritaoglu et al.[38] and Senior[47] modeled human appearance as a 2D temporal template with a probability map which records the likelihood that the pixel location (x, y) belongs to a person and some time-averaged version of color/brightness for that pixel. The model is used to find a match in an image or to support tracking after occlusion. However, this type of appearance model is not suitable for multi-camera tracking since the appearance models can be very different between camera views.

We describe an appearance color model as a function of height that assumes that people are standing upright and are dressed, generally, so that consistently colored or textured color regions are aligned vertically. Our color appearance model is a combination of [64] and [63], where pixels belonging to a particular person at a particular height are described by their color models, and the human body is partitioned into three major parts: head, torso, and bottom; the model is thus a cylindrical model having three height slices. Each body part has its own color model represented by a color distribution. To allow multimodal densities inside each part, we use kernel density estimation. Therefore, a body part does not need to be homogeneously colored as long as we obtain kernels belonging to all the colors for the body part model.

Let $M = \{\mathbf{c}_i\}_{i=1...N_M}$ be a set of pixels from a body part when $\mathbf{c}_i = (c_{i,1}, ..., c_{i,d})$ is a d-dimensional vector of color representation. Using Gaussian kernels and an independence assumption between color channels, the probability that an input pixel $\mathbf{c} = \{c_1, ..., c_d\}$ is from the model M can be estimated as

$$p_M(\mathbf{c}) = \frac{1}{N_M} \sum_{i=1}^{N_M} \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{1}{2}\left(\frac{c_j - c_{i,j}}{\sigma_j}\right)^2} \tag{4.1}$$

In order to handle illumination changes, we use normalized color ($r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, s = \frac{R+G+B}{3}$) or Hue-Saturation-Value (HSV) color space with a wider kernel for 's (brightness)' and 'Value' to cope with the higher variability of these lightness variables. On the other hand, chromaticity variables are more invariant to lighting conditions and so a kernel with a small variance (σ_j) can be used for more discrimination power. The lightness variable is still used to discriminate gray colored

objects. We used both the normalized color and HSV spaces in our experiments and observed similar performances.

Viewpoint-independent models can be obtained by viewing people from different perspectives using multiple cameras. A related calibration issue was addressed in [46, 48] since each camera output of the same scene point taken at the same time or different time may be slightly different depending on camera types and parameters. In order to handle the change in observed colors of an object a brightness transfer function from one camera to another is learned in the training phase in [46]. The correlation of visual information between different cameras is learned using Support Vector Regression and Hierarchical Principle Component Analysis to estimate the subject appearance across cameras in [48].

We used the same type of cameras and observed there is almost no difference between camera outputs except for different illumination levels (due to shadow and orientation effects) depending on the side of person's body; This level of variability is covered by our color model.

4.2 Multi-camera Multi-person Segmentation and Tracking

A framework for segmenting and tracking occluded people on a ground plane is presented in this section.

4.2.1 Foreground segmentation

Given image sequences from multiple overlapping views including people to track, we start by performing detection using BGS to obtain the foreground maps

in each view. The codebook-based background subtraction algorithm [62] is used. Its shadow removal capability increases the performance of segmentation and tracking.

Each foreground pixel in each view is labelled as the best matching person (i.e., the most likely class) by Bayesian pixel classification as in [63]. The posterior probability that an observed pixel $\mathbf{x}$ (containing both color $\mathbf{c}$ and image position (x, y) information) comes from person k is given by

$$P(k|\mathbf{x}) = \frac{P(k)P(\mathbf{x}|k)}{P(\mathbf{x})} \tag{4.2}$$

We use the color model in Eq.4.1 for the conditional probability $P(\mathbf{x}|k)$. The color model of the person's body part to be examined is determined by the information of $\mathbf{x}$'s position as well as the person's ground point and full-body height in the camera view (See Fig.4.1). The ground point and height are determined initially by the method defined subsequently in Sec.4.2.2.

The prior reflects the probability that person k occupies pixel $\mathbf{x}$. Given the ground point and full-body height of the person, we can measure $\mathbf{x}$'s height from the ground and its distance to the person's center vertical axis. The occupancy probability is then defined by

$$\begin{aligned} O_k(h_k(\mathbf{x}), w_k(\mathbf{x})) &= P[w_k(\mathbf{x}) < W(h_k(\mathbf{x}))] \\ &= 1 - \mathrm{cdf}_{W(h_k(\mathbf{x}))}(w_k(\mathbf{x})) \end{aligned} \tag{4.3}$$

where $h_k(\mathbf{x})$ and $w_k(\mathbf{x})$ are the height and width of $\mathbf{x}$ relative to the person k. $W(h_k(\mathbf{x}))$ is the person's height-dependent width and $\mathrm{cdf}_W(.)$ is the cumulative density function for W. If $\mathbf{x}$ is located at distance $W(h_k(\mathbf{x}))$ from the person's

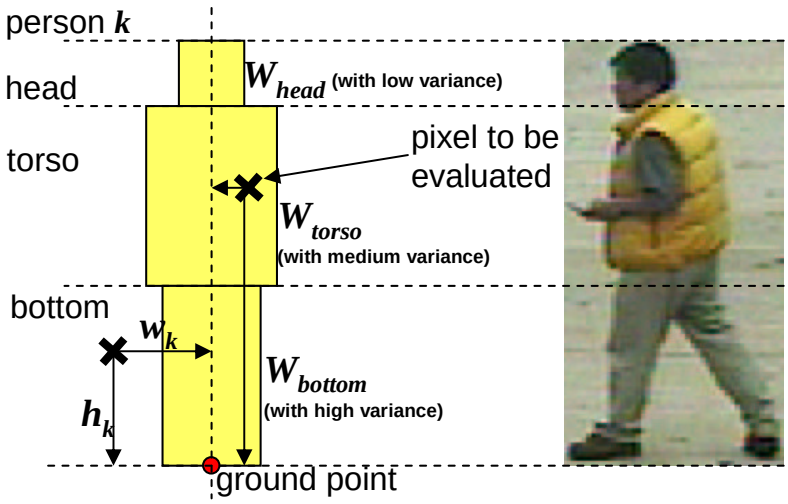


Figure 4.1: Illustration of appearance model for each body part. h_k and w_k are measured relative to the full height of the person.

center at a distance W, the occupancy probability is designed so that it will be exactly 0.5 (while it increases or decreases as $\mathbf{x}$ move towards or move away from the center).

The prior must also incorporate possible occlusion. Suppose that some person l has a lower ground point than a person k in some view. Then the probability that l occludes k depends on their relative positions and l's (probabilistic) width. Hence, the prior probability $P(k)$ that a pixel $\mathbf{x}$ is the image of person k, based on this occlusion model, is

$$P(k) = O_k(h_k, w_k) \prod_{g_y(k)<g_y(l)} (1 - O_l(h_l, w_l)) \tag{4.4}$$

where $g_y(k)$ is the y-location of the ground point of k and $\mathbf{x}$ is omitted for simplicity (i.e., $h_k = h_k(\mathbf{x})$ and $w_k = w_k(\mathbf{x})$).

The best class k^* is determined by maximum a posteriori (MAP) estimation:

$$k^* = \arg\max_k P(k)P(\mathbf{x}|k) \tag{4.5}$$

Figure 4.2: Detection of persons for initialization of the appearance model. The bounding boxes in the figures were created when the blobs are isolated before.

Finally, the foreground maps are segmented into the best matching persons based on their appearance models and occlusion information.

4.2.2 Model initialization and update

Full automatic tracking is enabled by initializing the human appearance model when a person is detected in a view by searching for isolated foreground blobs (See Fig.4.2). A heuristic method of human detection is used, which detects a head first by searching for significant peaks on the vertical projection histogram of the silhouettes and their corner vertices on the silhouette boundary.

The appearance color model in Eq.4.1 is updated by adding the classified pixel values into the sample list of the kernel estimator. Old samples will be forgotten gradually as new samples are added. The full-body height of a person is initialized upon model creation and is updated during segmentation. In some cases, fixing the average height scaled by the y-location of the ground point provides a robust height measurement when the segmentation is unreliable.

When the unclassified pixels (those having a probability in Eq.4.1 lower than a given threshold) constitute a connected component of non-negligible size, a new appearance model is created.

4.2.3 Multi-view integration

□ Ground plane homography

The segmented blobs across views are integrated to obtain the ground plane locations of people. The correspondence of a human across multiple cameras is established by the geometric constraints of planar homographies. A homography is a projective transformation represented as a nonsingular 3×3 matrix $\mathbf{H}$ defined only up to a scale, which is widely used for image warping or mosaicing.

Given a set of corresponding image points $\mathbf{p}$ and $\mathbf{q}$, belonging to a ground plane, from two camera views, the homography $\mathbf{H}$ satisfying $\mathbf{q} \equiv \mathbf{Hp}$ is recovered (Note that $\equiv$ denotes that the equality is in homogeneous coordinates, meaning that the left and right hand side are proportional).

Each pair of corresponding points gives two independent linear equations in the form of $\mathbf{q} \times \mathbf{Hp} = 0$. Since $\mathbf{H}$ has 8 unknowns, at least 4 point correspondences determine $\mathbf{H}$. We manually obtained those corresponding points, but there are several ways to do this automatically, such as [54]. When $\mathbf{H}$ is rewritten as

$$\mathbf{h} = (h_{11}, h_{12}, h_{13,} h_{21}, h_{22}, h_{23,} h_{31}, h_{32}, h_{33})^T,$$

N_{corres} point correspondence give $2N_{corres}$ linear equations, which result in a system of the form $\mathbf{Ah} = 0$. With the additional constraint that $\|\mathbf{h}\|^2 = 1$, the least squares solution of $\mathbf{h}$ is then given by the eigenvector corresponding to the smallest

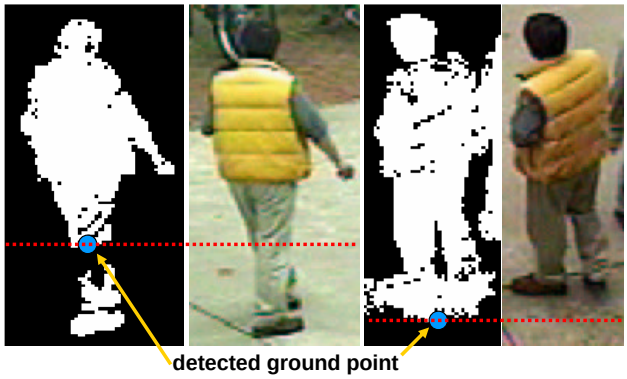


Figure 4.3: Wrong ground points were detected due to the broken segmentation and the shadow under the feet.

eigenvalue of the matrix $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

For N_V camera views, $N_V(N_V - 1)$ homography matrices can possibly be calculated for correspondence; but in order to reduce the computational complexity we instead reconstruct the top-view of the ground plane on which the hypotheses of peoples' locations are generated. So, for each image view, we need to obtain only two homographies relating the top-view with the image view.

□ Integration by vertical axes

Given the pixel classification results from Sec.4.2.1, a ground point of a person could be simply obtained by detecting the lowest point of the person's blob. Those image ground points of the person from all views can be mapped to the top-view plane using the homographies, and then could be averaged into a single ground point for tracking people's positions. However those ground points are not reliable due to the errors from background subtraction and segmentation, as shown in Fig.4.3.

We instead develop a localization algorithm that employs the center vertical axis of a human body, which can be estimated more robustly even with poor back-

ground subtraction [55]. Ideally, a person's body pixels are arranged more of less symmetrically about a person's central vertical axis. An estimate of this axis can be obtained by Least Mean Squares of the perpendicular distance between the body pixel and the axis. Alternatively, the Least *Median* Squares would also be more robust to outliers.

An interesting fact is that the homographic images of all the vertical axes of a person across different views intersect at (or are very close to) a single point (the location of that person on the ground) when mapped to the top-view (See [55]). In fact, even when the ground point of a person from some view is occluded, the top-view ground point integrated from all the views is obtainable only if the vertical axis is estimated correctly. This intersection point can be calculated by minimizing the perpendicular distances to the axes. Fig.4.4 depicts an example of reliable detection of the ground point from the segmented blobs of a person. The two vertical axes are mapped to the top-view and transferred back to each image view.

Let each axis L_i be parameterized by two points $\{(x_{i,1}, y_{i,1}), (x_{i,2}, y_{i,2})\}_{i=1...N_V}$. When mapped to the top-view by homography, we obtain $\{(\hat{x}_{i,1}, \hat{y}_{i,1}), (\hat{x}_{i,2}, \hat{y}_{i,2})\}_{i=1...N_V}$. The distance of a ground point (x, y) to the axis is written as

$$d\left((x, y), L_i\right) = \frac{|a_i x + b_i y + c_i|}{\sqrt{a_i^2 + b_i^2}} \tag{4.6}$$

where $a_i = \hat{y}_{i,1} - \hat{y}_{i,2}$, $b_i = \hat{x}_{i,2} - \hat{x}_{i,1}$, and $c_i = \hat{x}_{i,1}\hat{y}_{i,2} - \hat{x}_{i,2}\hat{y}_{i,1}$. The solution is the point that minimizes a weighted sum of square distances:

$$(x^*, y^*) = \arg\min_{(x,y)} \sum_{i=1}^{N_V} w_i^2 d^2((x, y), L_i) \tag{4.7}$$

The weight w_i is determined by the segmentation quality of the body blob of L_i

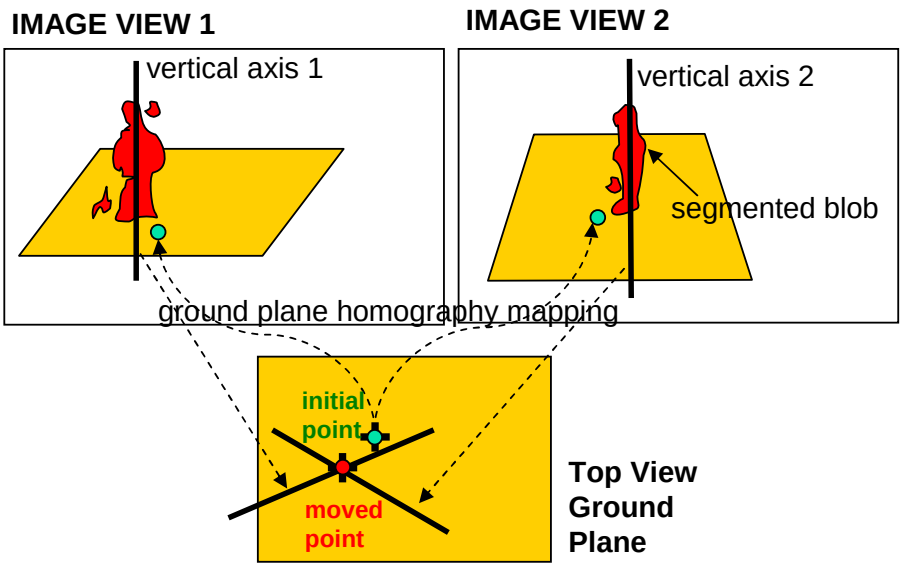


Figure 4.4: All vertical axes of a person across views intersect at (or are very close to) a single point when mapped to the top-view.

(We used the pixel classification score in Eq.4.2). The solution is easily calculated by solving the following linear system:

$$\begin{bmatrix} \sum_{i=1}^{N_V} \frac{w_i^2 a_i^2}{a_i^2+b_i^2} & \sum_{i=1}^{N_V} \frac{w_i^2 a_i b_i}{a_i^2+b_i^2} \\ \sum_{i=1}^{N_V} \frac{w_i^2 a_i b_i}{a_i^2+b_i^2} & \sum_{i=1}^{N_V} \frac{w_i^2 b_i^2}{a_i^2+b_i^2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -\sum_{i=1}^{N_V} \frac{w_i^2 a_i c_i}{a_i^2+b_i^2} \\ -\sum_{i=1}^{N_V} \frac{w_i^2 b_i c_i}{a_i^2+b_i^2} \end{bmatrix}$$

If a person is occluded severely by others in a view (i.e., the axis information is unreliable), the corresponding body axis from that view will not contribute to the calculation in Eq.4.7. When only one axis is found reliably, then the lowest body point along the axis is chosen.

To obtain a better ground point and segmentation result, we can iterate the segmentation and ground-point integration process until the ground point converges to a fixed location within a certain bound ϵ. That is, given a set of initial ground-point hypotheses of people, segmentation in Sec.4.2.1 is performed, and then newly moved ground points are obtained based on multi-view integration. These new

ground points are an input to the next iteration. 2-3 iterations gave satisfactory results for our data sets.

4.3 Extension to Multi-hypothesis Tracker

In this section, we extend our single-hypothesis tracker to one with multiple hypotheses. A single hypothesis tracker, while computationally efficient, can be easily distracted by occlusion or nearby similarly colored objects.

However, we need to deal with the challenge that as the number of targets and views increase, the state space of combination of targets' states increases exponentially. Additionally, the observation processes for visual tracking are typically very expensive.

The iterative segmentation-searching presented in Sec.4.2 can be naturally incorporated with a particle filtering framework. There are two advantages - (1) By searching for a person's ground point from a segmentation, a set of a few good particles can be identified, resulting in low computational costs, (2) Even if all the particles are away from the true ground point, some of them will move towards the true one as long as they are initially located nearby. This does not happen generally with particle filters, which need to wait until the target "comes to" the particles.

4.3.1 Overview of particle filter

In visual tracking, particle filtering is a Sequential Monte Carlo technique to apply a recursive Bayesian filter based on propagation of a sample set over time. Multiple hypotheses are kept to predict the position of the tracked object(s). In

recent years, particle filters have become popular for probabilistic tracking for non-linear/non-Gaussian models since they provide a robust framework with simplicity, generality and success in a wide range of challenging applications [61, 60].

In the sequential Bayesian filtering framework, the conditional density of the state variable given the measurements (observations) is propagated through the two step recursion:

$$\text{prediction} : p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int D(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) d\mathbf{x}_{t-1}$$

$$\text{update} : p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{M(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{\int M(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) d\mathbf{x}_t}$$

where $\mathbf{x}_t$[1] denotes the state of the tracked object and its probability density function is estimated from the sequence of measurement $\mathbf{z}_t$. The recursion requires the specification of a dynamic model $D(\mathbf{x}_t|\mathbf{x}_{t-1})$ that governs the state evolution, and a model $M(\mathbf{z}_t|\mathbf{x}_t)$ that reflects the likelihood of any state in terms of the current measurement.

While analytic methods for the recursion are not generally available due to the likelihood's non-linearity and multi-modality, particle filtering can nevertheless be used. The key idea of particle filtering is to approximate the probability distribution by a weighted sample set $S = \{(\mathbf{s}^{(n)}, \pi^{(n)})|n = 1...N\}$. Each sample, $\mathbf{s}$, represents one hypothetical state of the object, with a corresponding discrete sampling probability π, where $\sum_{n=1}^{N} \pi^{(n)} = 1$. Each element of the set is then weighted in terms of the observations and N samples are drawn with replacement, by choosing a particular sample with probability $\pi_t^{(n)} = M(\mathbf{z}_t|\mathbf{x}_t = \mathbf{s}_t^{(n)})$.

[1] $\mathbf{x}_t$ here is different from $\mathbf{x}_t$ in Sec.4.2.1 which denotes a pixel to be evaluated.

4.3.2 State space and dynamics

In our particle filtering framework, each sample of the distribution is simply given as

$$s = (x, y)$$

where x, y specify the ground location of the object in the *top-view*. For multi-person tracking, a state $\mathbf{s}_t = (\mathbf{s}_{1,t}, ..., \mathbf{s}_{N_p,t})$ is defined as a combination of N_p single-person states. Our state transition dynamic model is a random walk where a new predicted single-person state is acquired by adding a zero mean Gaussian with a covariance $\mathbf{\Sigma}$ to the previous state. Of course, the velocity $\dot{x}, \dot{y}$ or the size variable *height* and *width* can be added to the state space and then a more complex dynamic model can be applied if relevant.

4.3.3 Observation

Each person is associated with a reference color model $\mathbf{q}^\star$ which is obtained by histogram techniques [60]. The histograms are produced with the function $b(\mathbf{c}_i) \in \{1, ..., N_b\}$ that assigns the color vector $\mathbf{c}_i$ to the corresponding bin. We used the color model defined in Sec.4.1 to construct the histogram of the reference model in the normalized color or HSV space using N_b (e.g., $10 \times 10 \times 5$) bins to make the observation less sensitive to lighting conditions.

The histogram $\mathbf{q}(C) = \{q(u; C)\}_{u=1...N_b}$ of the color distribution of the sample set C is given by

$$q(u; C) = \eta \sum_{i=1}^{N_C} \delta[b(\mathbf{c}_i) - u] \tag{4.8}$$

where u is the bin index, δ is the Kronecker delta function, and η is a normalizing

constant ensuring $\sum_{u=1}^{N_b} q(u;C) = 1$. This model associates a probability to each of the N_b color bins.

If we denote $\mathbf{q}^\star$ as the reference color model and $\mathbf{q}$ as a candidate color model, $\mathbf{q}^\star$ is obtained from the stored samples of person k's appearance model as mentioned before while $\mathbf{q}$ is specified by a particle $\mathbf{s}_{k,t} = (x, y)$. The sample set C in Eq.4.8 is replaced with the sample set specified by $\mathbf{s}_{k,t}$. The top-view point (x, y) is transformed to an image ground point for a certain camera view v, $H_v(\mathbf{s}_{k,t})$, where H_v is a homography mapping the top-view to the view v. Based on the ground point, a region to be compared with the reference model is determined. The pixel values inside the region are drawn to construct $\mathbf{q}$. Note that the region can be constrained from the prior probability in Eq.4.4 including the occupancy and occlusion information (i.e., by picking pixels such that $P(k) > Threshold$, typically 0.5). In addition, as done in pixel classification, the color histograms are separately defined for each body part to incorporate the spatial layout of the color distribution. Therefore, we apply the likelihood as the sum of the histograms associated with each body part.

Then, we need to measure the data likelihood between $\mathbf{q}^\star$ and $\mathbf{q}$. The Bhattacharyya similarity coefficient is used to define a distance d on color histograms:

$$d[\mathbf{q}\star, \mathbf{q}(\mathbf{s})] = \left[1 - \sum_{u=1}^{N_b} \sqrt{q \star (u) q(u;\mathbf{s})}\right]^{\frac{1}{2}} \tag{4.9}$$

This distance between probability distributions is bounded within [0,1], and is an appropriate choice of measuring similarity of color histograms [52]. Thus, the likelihood of person k at view v is given by:

$$\pi_{v,k,t} \propto e^{\sum_{r=1}^{N_r} -\lambda d^2[\mathbf{q}_r^\star, \mathbf{q}_r(H_v(\mathbf{s}_{k,t}))]} \tag{4.10}$$

Note that since we get N_V likelihoods for a person across views, the actual likelihood of $\mathbf{s}_{k,t}$ is given by:

$$\pi_{k,t} = \Pi_{v=1}^{N_V} \pi_{v,k,t} \tag{4.11}$$

Finally, the weight of the particle of a concatenation of N_p person states is

$$\pi_k = \Pi_{k=1}^{N_p} \pi_{k,t}$$

4.3.4 The final algorithm

The algorithm below combines the particle filtering framework described before and the iterated segmentation-and-search into a final multi-view multi-target multi-hypothesis tracking (called M^3 Tracker).

Algorithm for M^{3}Tracker

I. From the "old" sample set $S_{t-1} = \{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}\}_{n=1,...,N}$ at time $t-1$, construct the new samples as follows:

II. **Prediction**: for $n = 1, ..., N$, draw $\tilde{\mathbf{s}}_t^{(n)}$ from the dynamics. **Iterate** Step III to IV for each particle $\tilde{\mathbf{s}}_t^{(n)}$.

III. **Segmentation & Search**
$\tilde{\mathbf{s}}_t = \{\tilde{\mathbf{s}}_{k,t}\}_{k=1...N_p}$ contains all persons' states. The superscript (n) is omitted through the Observation step.

i. **for** $v \leftarrow 1$ to N_V **do**

(a) For each person k, $(k = 1...N_p)$, transform the top-view point $\tilde{\mathbf{s}}_{k,t}$ into the ground point in view v by homography, $H_v(\tilde{\mathbf{s}}_{k,t})$

(b) perform segmentation on the foreground map in view v with the occlusion information according to Sec4.2.

end for

ii. For each person k, obtain the center vertical axes of the person across views, then integrate them on the top-view to obtain a newly moved point $\tilde{\mathbf{s}}_{k,t}^*$ as in Sec4.2.

iii. For all persons, if $\|\tilde{\mathbf{s}}_{k,t} - \tilde{\mathbf{s}}_{k,t}^*\| < \varepsilon$, then go to the next step. Otherwise, set $\tilde{\mathbf{s}}_{k,t} \leftarrow \tilde{\mathbf{s}}_{k,t}^*$ and go to Step III-i.

IV. **Observation**

i. **for** $v \leftarrow 1$ to N_V **do**

For each person k, estimate the likelihood $\pi_{v,k,t}$ in view $v4$ according to Eq.4.10. $\tilde{\mathbf{s}}_{k,t}$ needs to be transferred to view v by mapping through H_v for evaluation. Note that $\mathbf{q}_r(H_v(\tilde{\mathbf{s}}_{k,t}))$ is constructed only from the non-occluded body region.

end for

ii. For each person k, obtain the person likelihood $\pi_{k,t}$ by Eq.4.11.

iii. Set $\pi_t \leftarrow \Pi_{k=1}^{N_p} \pi_{k,t}$ as the final weight for the multi-person state $\tilde{\mathbf{s}}_t$.

V. **Selection**: Normalize $\{\pi_t^{(n)}\}_i$ so that $\sum_{n=1}^{N} \pi_t^{(n)} = 1$.
For $i = n...N$, sample index $a(n)$ from discrete probability $\{\pi_t^{(n)}\}_i$ over $\{1...N\}$, and set $\mathbf{s}_t^{(n)} \leftarrow \tilde{\mathbf{s}}_t^{a(n)}$.

VI. **Estimation**: the mean top-view position of person k is $\sum_{n=1}^{N} \pi_t^{(n)} \mathbf{s}_{k,t}^{(n)}$.

4.4 Experiments

We now presents experimental results obtained on outdoor multi-view sequences to illustrate the performance of our algorithm.

The sequences were captured at 15 frames/sec using three outdoor cameras placed on the side of a building at angular separation of 120 and 45 degrees approximately. Four human subjects walk together through the overlapping fields of view and moving around freely. The sequence is challenging in that three people are wearing similarly-colored clothes on their tops or bottoms and they sometimes approach close, making segmentation difficult. The number of particles (a combination of 4 single-person states) is 15, unless specified otherwise.

When angular separation is close to 180 degrees (visibility is maximized though), the intersection point of two vertical axes transformed to top-view may not be reliable because a small amount of angular perturbation make the intersected point move a lot. In our case, since camera 1 and 3 are placed at about 165 degrees, the problem was observed when the detection information from camera 2 is not available. This indicates that sensor placement can be an important factor for our tracker.

Fig.4.5 depicts the tracking results of all three views along with the particles on the top-view plane. The last row shows how the persons' vertical axes are intersecting on the top-view to obtain their ground points. When occlusion happens, the ground points being tracked are distracted a little but are recovered to the correct positions soon.

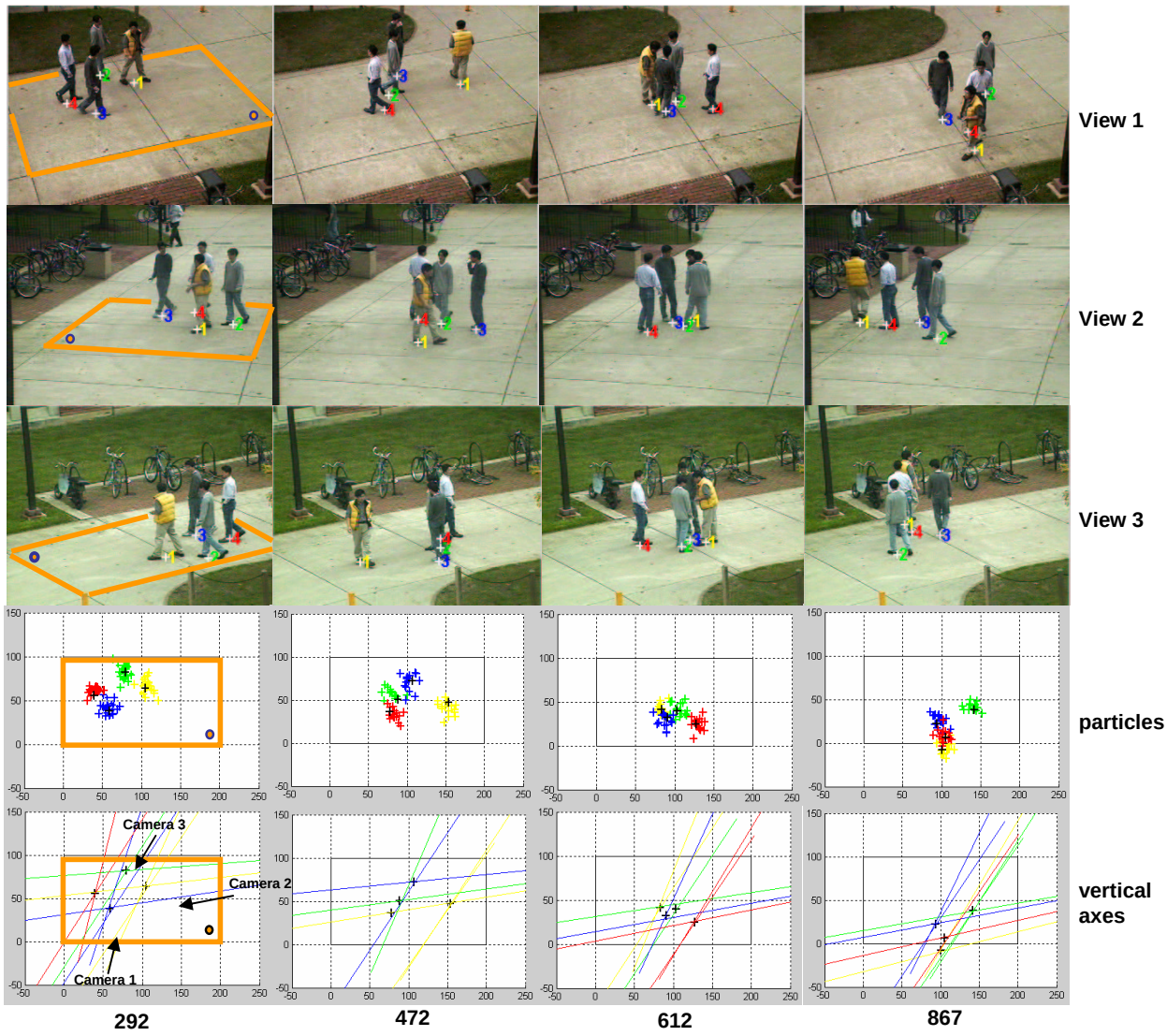


Figure 4.5: The proposed system tracks the ground positions of people over nearly 1000 frames. A small ball marker are overlaid on the resultant images of the frame 292 for easy finding of the camera orientations. Four representative frames are selected, which show the difficulty of tracking due to severe occlusion. Additionally, persons 2 and 3 are similar in their appearance colors. Note that, in the figures of 'vertical axes', the axis of a severely occluded person does not account for localization of the ground point.

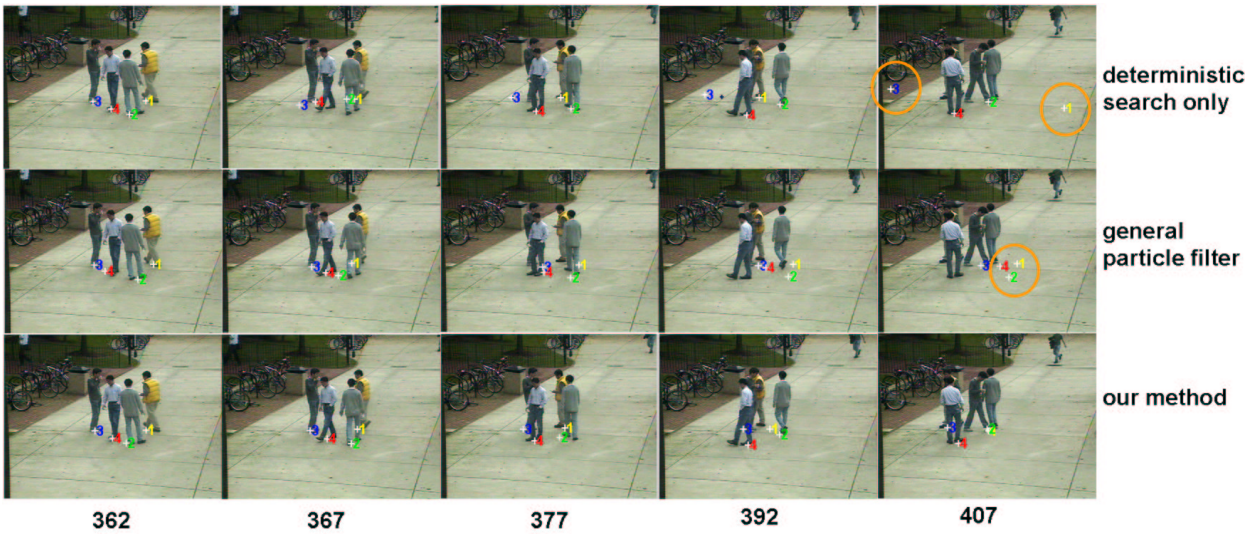


Figure 4.6: Comparison on three methods: While the deterministic search with a single hypothesis (persons 2 and 4 are good) and the general particle filter (only person 3 is good) fail in tracking all the persons correctly, our proposed method succeeds with a minor error. The view 2 was only shown here.

In order to demonstrate the advantage of our approach, single hypothesis (deterministic search only) tracker, general particle filter, and particle filter with deterministic search by segmentation (our proposed method) are compared in Fig.4.6. While the deterministic-search-only tracker keeps tracking for persons 2 and 4 correctly, it cannot recover the lost tracks, but the ground points (of persons 1 and 3) move away from the true positions. In the general particle filter, the particles cannot follow the true ground points well due to the insufficient observations by occlusion.

Another important feature of our framework is the ability to *segment* targets even though they are under severe occlusion. Some segmentation results are presented in Fig.4.7. Note that any blob-level image segmentation techniques are not employed but only pixel-level classification is performed in two or three iterations.

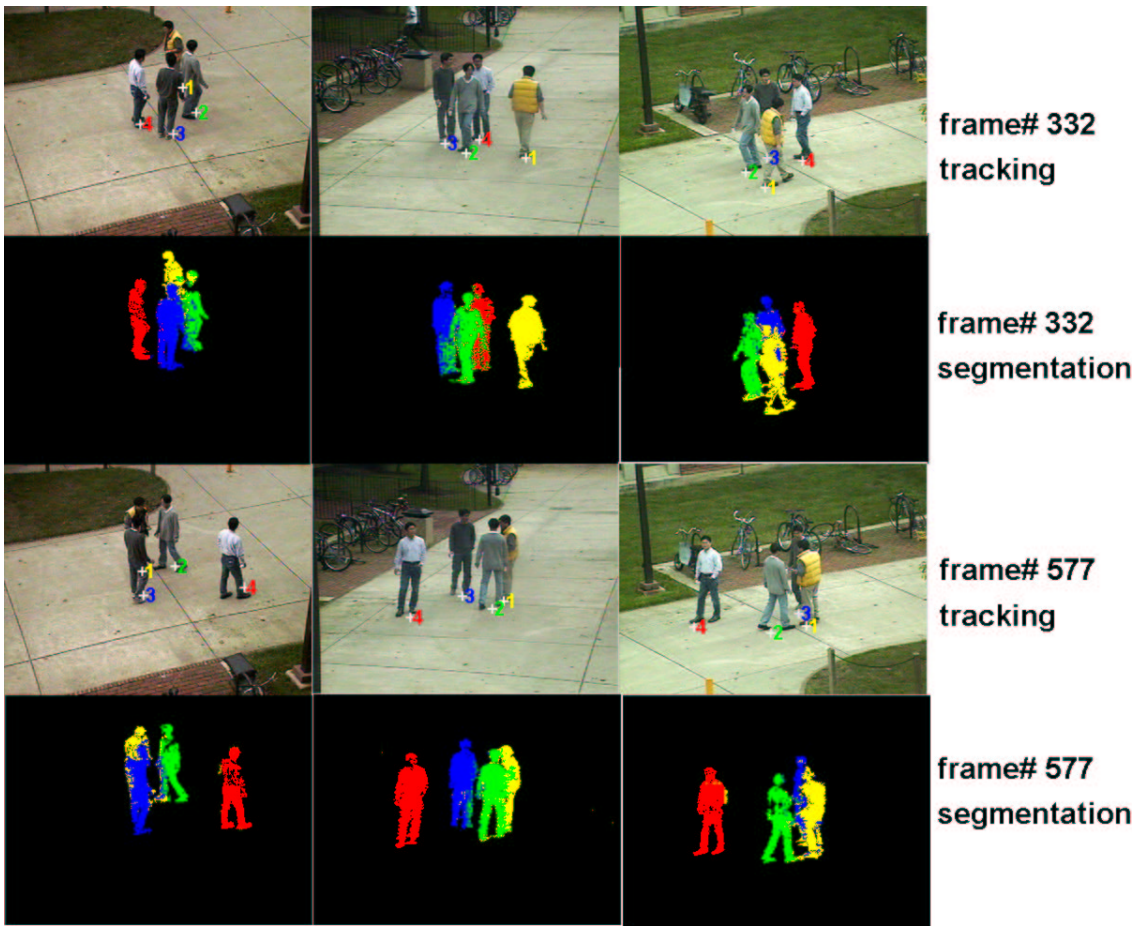


Figure 4.7: Segmentation results: Our method provides generally accurate blobs. The segmentation results which reflect occlusion can be obtained through the Bayesian pixel classification (Eq.4.2) based on the geometric constraints (Eq.4.4) determined by the locations of the ground points.

4.5 Conclusions

A framework to segment and track people on a ground plane is presented. Human appearance models are used to segment foreground pixels obtained from background subtraction. We developed a method to effectively integrate segmented blobs across views on a top-view reconstruction, with a help of ground plane homography. The multi-view tracker is extended efficiently to a multi-hypothesis framework (M^3Tracker) using particle filtering.

There are two important contributions:

1. To more precisely locate the ground location of a person, all center vertical axes of the person across views are mapped to the top-view plane (rather than compared within a pair of views) to find the intersection point.

2. To tackle the explosive state space due to multiple targets and views, an iterative segmentation-searching is incorporated into a particle filtering framework. By searching for a person's ground point from segmentation, a set of a few good particles can be identified, resulting in low computational costs. In addition, even if all the particles are away from the true ground point, some of them move towards the true one as long as they are located nearby.

We have illustrated results on challenging videos to show the usefulness of the proposed approach.

Chapter 5

A Fine-Structure Image/Video Quality Measure using Local Statistics

This chapter is organized as follows. Section 5.1 describes four video properties to be considered for video surveillance applications. Our quality measure is detailed in Section 5.2. Experimental results are presented along with the performance of a background subtraction algorithm in Section5.3. Section 5.4 provides conclusions and future work directions.

5.1 Video properties: Q1 - Q4

The following four properties should be considered for the performance of background modeling and foreground detection.

- Q1 - **noise**: are errors in the image acquisition process that result in pixel values that do not reflect the true intensities of the real scene. It could be introduced due to sensor sensitivity, electronic transmission, illumination fluctuation, camera vibration, etc. For modeling backgrounds, noise caused by pixel fluctuations should be properly modelled.

- Q2 - **contrast** (blur vs. sharpness): could be affected by camera optics, resolution, etc. Targets having low contrasts or blur effects are not easy to

detect or to obtain accurate boundaries for further analysis.

- Q3 - **color information**: represents how well color values are distributed over the intensity range.

- Q4 - **clipping**: Due to the range limitation of pixel value representation such as [0-255], pixels which are actually brighter or darker than these bounds are clipped. For clipped pixels, it is difficult to model backgrounds and detect foreground objects.

In this work, we focus on Q1 and Q2 which are directly related to performance of video surveillance systems.

5.2 Fine-structure image/video quality measure

In this section, A statistical local measure, CSAC (a Color version of SAC described in the next paragraph), is presented. Then, it is extended to FIQ (Fine-structure Image/video Quality) for image/video quality measurement. They incorporate both Q1 and Q2 described in Section 5.1. A median of FIQ's in a video can be used for a qualitative image/video quality measure.

SAC (center-Symmetric Auto-Correlation measure) for gray-scale images is defined by Eq.5.2 [74]. It is computed for center-symmetric pairs of pixels in a 3×3 neighborhood as in Fig.5.1-(left). μ and σ^2 denote the local mean and variance of the center-symmetric pairs. SCOV (center-symmetric covariance measure) is a measure of the pattern correlation as well as the local pattern contrast. Since SCOV is unnormalized, it is more sensitive to local sample variation. SAC is a "normalized"

Figure 5.1: (left) 3×3 neighborhood with 4 center-symmetric pairs of pixels, (right) original space station image

gray-scale invariant version of the covariance measure SCOV. The invariance makes SAC robust in the presence of local gray-scale variability or noise. The values of SAC are bounded between -1 and 1. For zero σ, SAC is defined as zero. For multi-band RGB-color imagery, CSAC can be defined by Eq.5.3.

$$\text{SCOV} = \frac{1}{4}\sum_{i}^{4}(p_i - \mu)(p_i' - \mu) \tag{5.1}$$

$$\text{SAC} = \frac{\text{SCOV}}{\sigma^2} \tag{5.2}$$

$$\text{CSAC} = \frac{1}{3}\sum_{c\in\{R,G,B\}}\frac{\text{SCOV}_c}{\sigma_c^2} \tag{5.3}$$

By inspecting CSAC histograms in Fig.5.2, one can observe the effects of noise and blur. Each histogram shows the CSAC distribution of all the pixels in each image. Gaussian noise and blur filters (of size 5×5) have been applied to the original image in Fig.5.1-(right) to obtain noisy and blurry versions used for generating the CSAC histograms in Fig.5.2.

Most CSAC's of the original image are negative-symmetric, which corresponds to edge-like patterns. CSAC's close to zero reflect noisy patterns. Positive-symmetric patterns correspond to details in an image. The distribution of the GaussianNoise

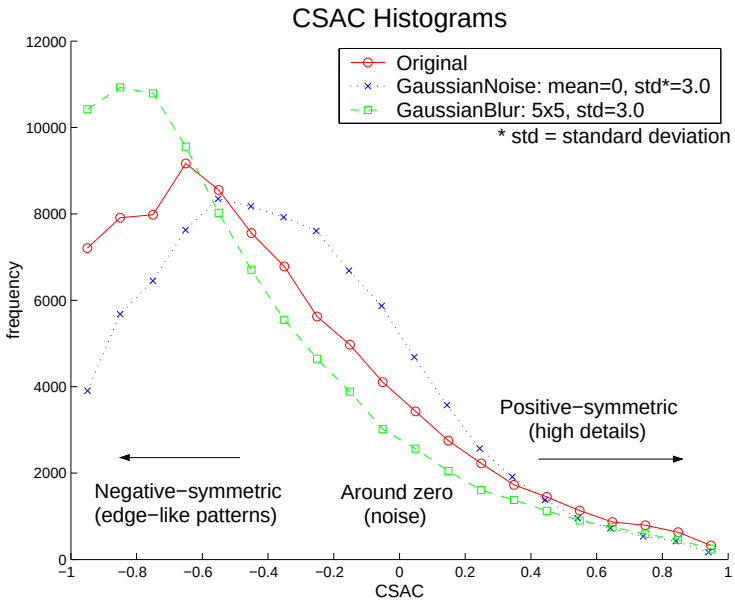


Figure 5.2: CSAC histograms of original, noise, and blur images

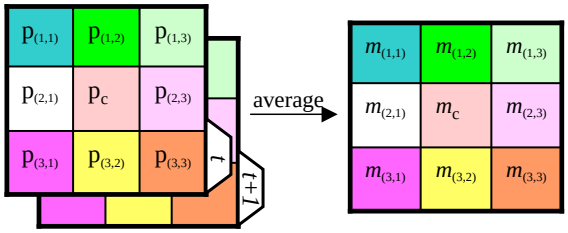


Figure 5.3: two 3×3 boxes and their average.

image is shifted toward zero. The GaussianBlur image reduces noise as seen from its histogram around zero compared with the original one. Note that both decrease high details (positive-symmetric) in the original image.

However, we have two issues about a CSAC measure - (1) It cannot be used for a quantitative image quality measure, (2) It represents the quality of a single image, not a video or an image sequence.

In this regard, as an alternative of CSAC, a variance ratio (VR) for gray-scale images can be defined by Eq.5.4 where WVAR is the within variance over space and BVAR is the between variance over time. For two consecutive frames of frame

number t and $t+1$, WVAR and BVAR for each pixel are defined by Eq.5.5 where N is 8-neighbors of an interesting center pixel c. $m_{(x,y)}$ is an average of two pixels $p_{(x,y,t)}$ and $p_{(x,y,t+1)}$ (See Fig.5.3).

$$\mathrm{VR} = \frac{\mathrm{WVAR}}{\mathrm{BVAR}} \tag{5.4}$$

$$\begin{aligned} \mathrm{WVAR} &= \tfrac{1}{8} \sum_{(x,y)\in N} (m_{(x,y)} - m_c)^2 \\ \mathrm{BVAR} &= \tfrac{1}{9} \sum_{(x,y)\in N\cup\{c\}} (p_{(x,y,t)} - p_{(x,y,t+1)})^2 \end{aligned} \tag{5.5}$$

VR can be viewed as signal-to-noise ratio since WVAR is a measure of local pattern contrast and BVAR is measure of noise over time. WVAR can also be defined in terms of a second derivative-like measure or a laplacian to measure fine structure quality better. A laplacian version is presented in Eq.5.6. Let's call the square root version using Eq.5.6 as FIQ (fine-structure image/video quality) as in Eq.5.7. A FIQ measure is a normalized laplacian and a ratio of within and between frame variation. Note that FIQ is not ranged from 0 to 1 since WVAR and BVAR have different scaling factors. For color imagery, FIQ can be defined as an average of FIQ's of all color channels.

$$\mathrm{WVAR} = \left[\nabla^2\right]^2 = \left[\frac{1}{8} \sum_{(x,y)\in N} (m_{(x,y)} - 8m_c)\right]^2 \tag{5.6}$$

$$\mathrm{FIQ} = \frac{\nabla^2}{\sqrt{\mathrm{BVAR}}} \tag{5.7}$$

While adding noise makes both WVAR and BVAR increase, BVAR is increased relatively more than WVAR. In blurred images, WVAR and BVAR are decreased.

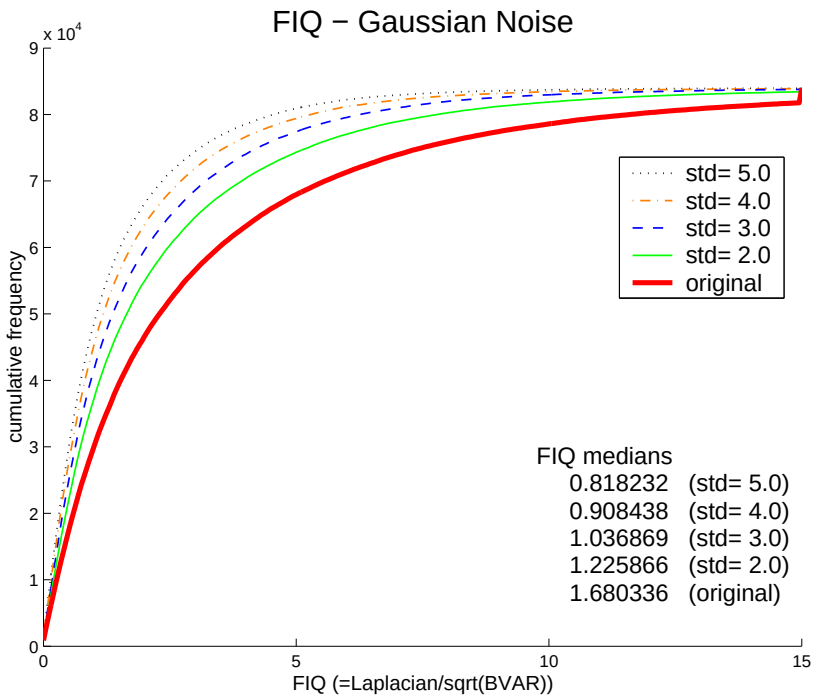


Figure 5.4: Cumulative FIQ histograms - GaussianNoise

But WVAR is decreased relatively more than BVAR since blur filters are spatial low-pass filters. All these facts make FIQ measures of 'noise' and 'blur' images smaller than that of an original image. Fig.5.4,5.5 shows the cumulative FIQ histograms for the image tested in Fig.5.2. Once all FIQ's over space (all pixels) and time (all frames) are obtained, a median of those FIQ's can be used for a quantitative image/video quality measure since medians are robust statistics not affected by outliers. We use the FIQ median as our image/video quality measure. The medians for the original, GaussianNoise (std=3.0), and GaussianBlur (5×5, std=3.0) images are 1.680, 1.037, and 1.053 respectively. These values are pretty distinctive for quality measurement. More noise and blur make these FIQ medians smaller.

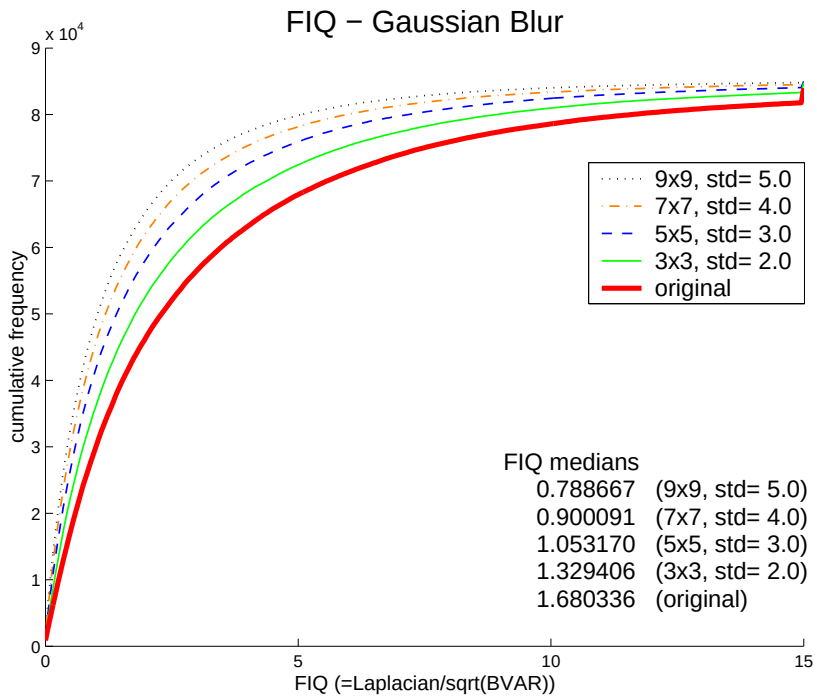


Figure 5.5: Cumulative FIQ histograms - GaussianBlur

5.3 Experimental Results

Our image/video quality measurement program produces FIQ histograms in Fig.5.4,5.5 as well as a text output which contains a FIQ median (related to Q1 and Q2), color entropy (Q3), and clipping information (Q4), which are shown below (For color images, an average value over all the color-bands is reported.).

```
================================================
      Image/Video Quality Statistics
================================================
-image.height = 240
-image.width = 360
-number of frames = 50
-clipping low_bound and high_bound = [20, 235]

[ Original ]
        clipped_low  =    0.16%
        clipped_high =    1.46%
        non-clipped  =   98.38%
            entropy    =  7.423942
            FIQ median =  1.680336

[ GaussianNoise: mean=0.0, std=3.0 ]
```

```
        clipped_low  =    0.17%
        clipped_high =    1.47%
        non-clipped  =   98.36%
            entropy    =  7.444018
            FIQ median =  1.036869

[ GaussianBlur: 5x5, std=3.0 ]
        clipped_low  =    0.00%
        clipped_high =    0.59%
        non-clipped  =   99.41%
            entropy    =  7.357112
            FIQ median =  1.053170
=======================================
```

Pixel values lower or higher than the given clipping bounds are classified as 'clipped'. The reason why we have a range smaller than [0,255] is that actual clipping effects occur before a pixel's brightness reaches the min or max limit, 0 and 255. An entropy and a FIQ median are measured only for non-clipped pixels. To measure color information, for each color band, an entropy H is obtained by $H = -\sum_{i=low\_bound}^{high\_bound} p_i \log p_i$ where p_i is the probability of the gray-level i.

The capability of detecting moving foreground objects from a video sequence captured using a static camera is a typical first step in visual surveillance. It is called 'background subtraction'. We tested a codebook-based background subtraction algorithm in [75] on the image sequences used in Section 5.2. The original image sequence is filtered by GaussianNoise or GaussianBlur. Detection performance is centralized around errors of false positive (FP) and false negative (FN).

As presented in Table.5.1, adding noise increases FP's while FN's are relatively stable. In the GaussianBlur case in Table.5.2, blurring increases FN's dramatically which means that there are a lot of miss detection. Note that it reduces FP's, but it does not affect its detection performance since the background area is much larger than the foreground area. Even, some spot FP's in the original image can

	FIQ median	FP	FN
original	1.680	3069	2945
std = 2.0	1.226	3325	2844
std = 3.0	1.037	3659	2864
std = 4.0	0.908	4169	2754
std = 5.0	0.818	5266	2746

Table 5.1: Errors for GaussianNoise images

	FIQ median	FP	FN
original	1.680	3069	2945
3×3, std = 2.0	1.329	889	3332
5×5, std = 3.0	1.053	710	3553
7×7, std = 4.0	0.900	638	3870
9×9, std = 5.0	0.789	572	4109

Table 5.2: Errors for GaussianBlur images

be eliminated by simple post-processing. In overall, it is shown that images (or a video) having lower FIQ's achieve poor performance.

5.4 Conclusions and Future Work

A fine-structure image/video quality measure has been presented. It has been shown that the proposed metric reflect image degradation well in terms of noise and blur. For video surveillance tasks such as background subtraction, FIQ can be used to measure the quality of video. Testing on a background subtraction algorithm supports its usefulness.

Future research directions include the followings:

- Quality measurement for different system settings such as different cameras, illuminations, focuses, or exposures. An operator can tune a surveillance system to get better performance.

- Applying our techniques to other video surveillance tasks like tracking and recognition.
- Testing high-shutter videos. We observed that an image sequence taken at a high shutter speed gives very accurate foreground silhouettes.
- Automatic parameter estimation for background subtraction algorithms. We already used BVAR to estimate a sampling bandwidth of background modeling. This is very important for practical use of video surveillance systems.

Chapter 6

Conclusions

In Chapter 2, our new adaptive background subtraction algorithm, which is able to model a background from a long training sequence with limited memory, works well on moving backgrounds, illumination changes (using our color distortion measures), and compressed videos having irregular intensity distributions. It has other desirable features - unconstrained training and layered modeling/detection. Comparison with other multimode modeling algorithms shows that the codebook algorithm has good properties on several background modeling problems.

In Chapter 3, we presented a perturbation method for measuring sensitivity of BGS algorithms. The PDR method does not require foreground targets in videos or knowledge of actual foreground distributions. PDR analysis does not consider all possible background or foreground distributions; it considers only those relevant to one video, scene and camera. It assumes that the foreground, when it has small contrast to the background locally, has a distribution similar in form to the background, but shifted or perturbed. PDR analysis has two advantages over the commonly used ROC analysis: (1) It does not depend on knowing foreground distributions, (2) It does not need the presence of foreground targets in the video in order to perform the analysis, while this is required in the ROC analysis. Because of these consider-

ations, PDR analysis provides practical general information about the sensitivity of algorithms applied to a given video scene over a range of parameters and FA-rates. In ROC curves, we obtain one detection rate for a particular FA-rate for a particular foreground and contrast.

In Chapter 4, a framework to segment and track people on a ground plane is presented. Human appearance models are used to segment foreground pixels obtained from background subtraction. We developed a method to effectively integrate segmented blobs across views on a top-view reconstruction, with a help of ground plane homography. The multi-view tracker is extended efficiently to a multi-hypothesis framework (M^3 Tracker) using particle filtering. To locate more precise ground location of a person, all center vertical axes of the person across views are mapped to the top-view plane (rather than compared within a pair of views) to find the intersection point. This is quite useful because background subtraction and segmentation are not always reliable due to noise, illumination changes, etc. To tackle with the explosive state space due to multiple targets and views, the iterative segmentation-searching is incorporated with a particle filtering framework. By searching the ground point from segmentation, a set of a few good particles can be identified, resulting in low computational costs. In addition, even if all the particles are away from the true ground point, some of them are to move towards to the true one as long as they are located nearby. This good feature does not happen to general particle filters. They need to wait until the target comes to the particles.

In Chapter 5, a fine-structure image/video quality measure has been presented. It has been shown that the proposed metric reflect image degradation well in terms

of noise and blur. For video surveillance tasks such as background subtraction, FIQ can be used to measure the quality of video. Testing on a background subtraction algorithm supports its usefulness.

BIBLIOGRAPHY

[1] C.R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, no. 7, pp. 780-785, 1997.

[2] T. Horprasert, D. Harwood, and L.S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection", *IEEE Frame-Rate Applications Workshop*, Kerkyra, Greece, 1999.

[3] C. Stauffer and W.E.L. Grimson, "Adaptive background mixture models for real-time tracking", *IEEE Int. Conf. Computer Vision and Pattern Recognition*, Vol. 2, pp. 246-252, 1999.

[4] D. S. Lee, J. J. Hull, and B. Erol , "A Bayesian Framework for Gaussian Mixture Background Modeling", *IEEE International Conference on Image Processing*, 2003.

[5] M. Harville, "A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models", *European Conf. Computer Vision*, Vol. 3, pp. 543-560, 2002.

[6] O. Javed, K. Shafique, M. Shah, "A Hierarchical Approach to Robust Background Subtraction using Color and Gradient Information ", *IEEE Workshop on Motion and Video Computing (MOTION'02)*, 2002.

[7] F. Porikli and O. Tuzel, "Human Body Tracking by Adaptive Background Models and Mean-Shift Analysis", *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-ICVS)*, 2003.

[8] M. Cristani, M. Bicego, V. Murino, "Integrated region- and pixel-based approach to background modelling", *Proc. IEEE Workshop on Motion and Video Computing*, 2002.

[9] A. Elgammal, D. Harwood, and L.S. Davis, "Non-parametric model for background subtraction," *European Conf. Computer Vision*, Vol. 2, pp. 751-767, 2000.

[10] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance", *Int. Conf. Computer Vision*, pp. 255-261, 1999.

[11] A. Mittal and N. Paragios, "Motion-based Background Subtraction Using Adaptive Kernel Density Estimation", *IEEE Conference in Computer Vision and Pattern Recognition*, 2004.

[12] N. Paragios and V. Ramesh, "A MRF-based Real-Time Approach for Subway Monitoring", *IEEE Conference in Computer Vision and Pattern Recognition*, 2001.

[13] D. Wang, T. Feng, H. Shum, S. Ma, "A Novel Probability Model for Background Maintenance and Subtraction", *The 15th International Conference on Vision Interface*, 2002.

[14] J. Zhong, S. Sclaroff, "Segmenting Foreground Objects from a Dynamic Textured Background via a Robust Kalman Filter", *IEEE International Conference on Computer Vision*, 2003.

[15] A. Monnet, A. Mittal, N. Paragios, V. Ramesh, "Background Modeling and Subtraction of Dynamic Scenes", *IEEE International Conference on Computer Vision*, 2003.

[16] A. Amer, E. Dubois, and A. Mitiche, "Real-time system for high-level video representation: application to video surveillance", *Proc. SPIE Int. Symposium on Electronic Imaging, Conf. on Visual Communication and Image Processing (VCIP)*, 2003.

[17] M. Greiffenhagen, V. Ramesh, D. Comaniciu, H. Niemann, "Statistical modeling and performance characterization of a real-time dual camera surveillance system", *Proc. Int. Conf. Computer Vision and Pattern Recognition*, Vol. 2, pp. 335-342, 2000.

[18] T. Kohonen, "Learning vector quantization", *Neural Networks*, Vol. 1, pp. 3-16. 1988.

[19] B.D. Ripley, "Pattern Recognition and Neural Networks", *Cambridge University Press*, 1996.

[20] Antoine Monnet, Anurag Mittal, Nikos Paragios and Visvanathan Ramesh, "Background Modeling and Subtraction of Dynamic Scenes", *IEEE International Conference on Computer Vision (ICCV)*, Nice, France, Oct 2003.

[21] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J.M. Buhmann, "Topology free hidden Markov models: application to background modeling", In *IEEE International Conference on Computer Vision*, Volume: 1, 2001, Page(s): 294 -301.

[22] A. Mittal and L. S. Davis, "M2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene Using Region-Based Stereo", *Proc. of European Conf. on Computer Vision*, pp. 18-33, 2002.

[23] K. Garg and S.K. Nayar, "Detection and Removal of Rain from Videos", *IEEE Computer Vision and Pattern Recognition (CVPR)*, Washington, July 2004.

[24] Tao Zhao and Ram Nevatia, "Tracking Multiple Humans in Crowded Environment", *Proc IEEE Conf on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[25] Y. Ren, C. Chua, and Y. Ho, "Statistical background modeling for non-stationary camera", *Pattern Recognition Letters*, vol. 24, no. 1-3, pp. 183-196, January 2003.

[26] Eric Hayman and Jan-Olof Eklundh, "Statistical Background Subtraction for a Mobile Observer", *IEEE International Conference on Computer Vision*, 2003.

[27] Dirk Walther, Duane R.Edgington, and Christof Koch, "Detection and Tracking of Objects in Underwater Video", *IEEE International Conference on Computer Vision and Pattern Recognition*, 2004.

[28] Yasuyuki Matsushita, Ko Nishino, Katsushi Ikeuchi, Masao Sakauchi, "Illumination Normalization with Time-Dependent Intrinsic Images for Video Surveillance", *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(10): 1336-1347 (2004).

[29] James W. Davis, Vinay Sharma, "Robust Background-Subtraction for Person Detection in Thermal Imagery", *Joint IEEE Workshop on Object Tracking and Classification Beyond the Visible Spectrum*, 2004.

[30] Hulya Yalcin, Michael J. Black, Ronan Fablet, "The Dense Estimation of Motion and Appearance in Layers", *Proc IEEE Conf on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[31] Qifa Ke and Takeo Kanade, "A Robust Subspace Approach to Layer Extraction", *IEEE Workshop on Motion and Video Computing (Motion 2002)*, pages 37-43, 2002.

[32] Philip H.S. Torr, Richard Szeliski, P. Anandan, "An Integrated Bayesian Approach to Layer Extraction from Image Sequences:, *IEEE Trans. Pattern Anal. Mach. Intell.* 23(3): 297-303, (2001).

[33] Paul Smith, Tom Drummond, Roberto Cipolla, "Layered Motion Segmentation and Depth Ordering by Tracking Edges", *IEEE Trans. Pattern Anal. Mach. Intell.*, April 2004.

[34] Brendan J. Frey, Nebojsa Jojic, Anitha Kannan, "Learning Appearance and Transparency Manifolds of Occluded Objects in Layers", *IEEE International Conference on Computer Vision and Pattern Recognition*, 2003.

[35] Arno Schodl and Irfan A. Essa, "Depth layers from occlusions", *IEEE International Conference on Computer Vision and Pattern Recognition*, 2001.

[36] Yue Zhou and Hai Tao, " Background Layer Model for Object Tracking through Occlusion", *IEEE International Conf. on Computer Vision, ICCV'03*, pp. 1079-1085, 2003.

[37] D. Harwood, M. Subbarao, H. Hakalahti, and L.S. Davis, "A New Class of Edge-Preserving Smoothing Filters", *Pattern Recognition Letters*, 6:155-162, 1987.

[38] I. Haritaoglu, D. Harwood, L.S. Davis, "W^4: real-time surveillance of people and their activities" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume: 22, Issue: 8, Aug 2000, Page(s): 809 -830.

[39] H.-D. Cheng, X.-H. Jiang, Y. Sun, J. Wang, "Color image segmentation: advances and prospects",, *Pattern Recognition*, 34 (12), 2001.

[40] G. Scotti, L. Marcenaro, C. Regazzoni, "A S.O.M. based algorithm for video surveillance system parameter optimal selection", *IEEE Conference on Advanced Video and Signal Based Surveillance*, 2003.

[41] X. Gao, T.E. Boult, F. Coetzee, V. Ramesh, "Error Analysis of Background Adaption," *Proc. Int. Conf. Computer Vision and Pattern Recognition*, pp. 503-510, 2000.

[42] P. Courtney and N. Thacker, "Performance characterisation in computer vision: The role of statistics in testing and design", In *J. Blanc-Talon, D. Popescu, eds.:*

Imaging and Vision Systems: Theory, Assessment and Applications, NOVA Science Books (2001).

[43] A.K. Jain, R.P.W. Duin, and J. Mao, "Statistical Pattern Recognition: A Revew", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume: 22, No. 1, Jan 2000.

[44] K. Bowyer, C. Kranenburg, and S. Dougherty, "Edge detector evaluation using empirical ROC curves" In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999, Volume: 1.

[45] Zhuowen Tu, Song-Chun Zhu, "Image segmentation by data-driven Markov chain Monte Carlo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 24, Issue 5, May 2002, Page(s):657 - 673.

[46] Omar Javed, Khurram Shafique and Mubarak Shah, "Appearance Modeling for Tracking in Multiple Non-overlapping Cameras," *IEEE CVPR 2005*, San Diego, June 20-26.

[47] A. W. Senior, "Tracking with Probabilistic Appearance Models," in *proceedings ECCV workshop on Performance Evaluation of Tracking and Surveillance Systems*, 1 June 2002, pp 48-55.

[48] Chang, T.H., Gong, S., Ong, E.J., "Tracking Multiple People Under Occlusion Using Multiple Cameras," BMVC 2000.

[49] Jinman Kang, Isaac Cohen, and Gerard Medioni, "Multi-Views Tracking Within and Across Uncalibrated Camera Streams", *Proceedings of the ACM SIGMM 2003 Workshop on Video Surveillance*, 2003.

[50] Javed O, Rasheed Z, Shafique K and Shah M, "Tracking Across Multiple Cameras With Disjoint Views," *The Ninth IEEE International Conference on Computer Vision*, Nice, France, 2003.

[51] D. Comaniciu, P. Meer, "Mean Shift: A Robust Approach toward Feature Space Analysis," *IEEE Trans. Pattern Analysis Machine Intell.*, Vol. 24, No. 5, 603-619, 2002.

[52] D. Comaniciu, V. Ramesh, P. Meer, "Real-Time Tracking of Non-Rigid Objects using Mean Shift," *IEEE Conf. on Comp. Vis. and Pat. Rec.*, 2000.

[53] Tao Zhao; Nevatia, R., "Bayesian human segmentation in crowded situations," *CVPR*, June 2003.

[54] Chris Stauffer, Kinh Tieu. "Automated multi-camera planar tracking correspondence modeling," *CVPR*, vol. 01, no. 1, p. 259, 2003.

[55] Min Hu, Jianguang Lou, Weiming Hu, Tieniu Tan, "Multicamera correspondence based on principal axis of human body," *International Conference on Image Processing*, 2004.

[56] S. Arulampalam, S. Maskell, N. J. Gordon, and T. Clapp, "A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking", *IEEE Transactions of Signal Processing*, Vol. 50(2), pages 174-188, February 2002.

[57] J. Deutscher, A. Blake and Ian Reid, "Articulated Body Motion Capture by Annealed Particle Filtering," *CVPR* 2000.

[58] J. Sullivan and J. Rittscher, "Guiding random particles by. deterministic search," *Proc. of ICCV*, 2001.

[59] Caifeng Shan, Yucheng Wei, Tieniu Tan, Frederic Ojardias, "Real Time Hand Tracking by Combining Particle Filtering and Mean Shift," S*ixth IEEE International Conference on Automatic Face and Gesture Recognition*, 2004.

[60] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," *ECCV* 2002.

[61] Michael Isard and Andrew Blake, "CONDENSATION – conditional density propagation for visual tracking," *Int. J. Computer Vision*, 29, 1, 5–28, 1998.

[62] Kyungnam Kim, Thanarat H. Chalidabhongse, David Harwood, and Larry Davis, "Real-time foreground-background segmentation using codebook model,", *Real-Time Imaging*, Volume 11, Issue 3, June 2005, Pages 172-185.

[63] Anurag Mittal and Larry S. Davis, "M_2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene," *International Journal of Computer Vision*, Vol. 51 (3), Feb/March 2003.

[64] A. Elgammal and L. S. Davis, "Probabilistic Framework for Segmenting People Under Occlusion", *IEEE International Conference on Computer Vision*, Vancouver, Canada July 9-12, 2001.

[65] M. Moore, S. Mitra, J. Foley, "Defect visibility and content importance implications for the design of an objective video fidelity metric", IEEE International Conference on Image Processing, June 2002.

[66] A.B. Watson, J. Malo, "Video quality measures based on the standard spatial observer", IEEE International Conference on Image Processing, June 2002.

[67] N.B. Nill, B.H. Bouzas, "Objective Image Quality Measure Derived From Digital Image Power Spectra", Optical Engineering, Vol. 31, No. 4, pp. 813-825, 1992.

[68] Z. Wang, A.C. Bovik, H.R. Sheikh and E.P. Simoncelli, "Image quality assessment: From error measurement to structural similarity," IEEE Transactions on Image Processing, vol. 13, no. 1, Jan. 2004.

[69] Video Quality Expert Group, http://www.vqeg.org/.

[70] A.M. Eskicioglu, P.S. Fisher, "Image Quality Measures and Their Performance", IEEE Transactions on Communications, 43(12), 2959-2965 (1995).

[71] İ. Avcibaş, B. Sankur, K. Sayood, "Statistical Evaluation of Image Quality Measures", Journal of Electronic Imaging, Vol. 11, pp. 206-223, April, 2002.

[72] P. Marziliano, F. Dufaux, S. Winkler, T. Ebrahimi, "A no-reference perceptual blur metric", IEEE International Conference on Image Processing, June 2002.

[73] D.S. Turaga, Y. Chen, J. Caviedes, "No reference PSNR estimation for compressed pictures", IEEE International Conference on Image Processing, June 2002.

[74] D. Harwood D, T. Ojala, M. Pietikainen, S. Kelman, and L.S. Davis, "Texture classification by center-symmetric auto-correlation, using Kullback discrimination of distributions". *Pattern Recognition Letters*, 1995, 16:1-10.

[75] T. H. Chalidabhongse, K. Kim, D. Harwood and L. Davis, "A Perturbation Method for Evaluating Background Subtraction Algorithms", Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS), 2003.